ISSN 0868-6157

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

5¼" disk version

Microsoft. MS-DOS.
Operating System Version 4.01

For Personal Computers 100% Compatible
with IBM Personal Computers

Includes Microsoft GW-BASIC Interpreter

Безопасность компьютерных систем

Inclndes
Microsoft Works
Demonstration Disk
and a $10.00
Rebate Offer

Not for retail sale except with a computer system.

Microsoft

10'91

ВИКТОРИЯ – ЭТО ВАША ПОБЕДА

Работать с "Викторией" проще, чем два пальца показать

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

10'91

Совместное советско-американское предприятие «СОВАМИНКО»

КОМПЬЮТЕР ПРЕСС

ОБОЗРЕНИЕ ЗАРУБЕЖНОЙ ПРЕССЫ

Адрес редакции:
113093, г.Москва, аб.ящик 37
Факс: 200-22-89
Телефоны для справок:
491-01-53, 420-83-80.
E-mail:
postmaster@Computerpress.msk.su

Сегодня мы предлагаем вашему вниманию цикл статей, посвященных проблемам обеспечения безопасности компьютерных систем. Этот специальный выпуск – не сборник рецептов, как избежать неприятностей, а скорее общая постановка вопроса. Проблемы безопасности стоят на повестке дня. В стране, как снежный ком, растет количество бирж, брокерских контор, коммерческих банков, различных фирм. Все они заинтересованы в оперативном обмене информацией. Причем, всем им нужно, чтобы их информация не попадала в третьи руки, потому что часто эта информация – почти что живые деньги.

К сожалению, пока что у нас берегутся в основном государственные да военные секреты, тщательно прячутся партийные тайны, но большинство людей от только что народившегося отечественного бизнеса не особенно утруждают себя защитой коммерческой информации. Видимо, потребуется два-три громких ска́ндала с миллиардными убытками, чтобы заинтересованные люди начали задумываться об этих проблемах.

Мы хотим обратить ваше внимание на актуальность этой проблемы. Мы хотим призвать тех, кто владеет необходимыми знаниями и информацией, помочь в ее решении.

Сдано в набор 25.09.91. Подписано к печати 10.10.91. Формат 84x108/16. Печать офсетная.
Усл.печ.л.8,4+0,32 (обл.). №033. Тираж 100 000 экз. (1 завод–55 000). Заказ 2246. Цена 3 р. 15 к.

Типография издательства «Калининградская правда»
236000, г.Калининград, ул.Карла Маркса, 18

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

Сегодня мы публикуем цикл статей, посвященных проблемам обеспечения безопасности информации, при обработке ее в компьютерах. Это не сборник рецептов, как избежать неприятностей. Этот специальный выпуск скорее преследует цель общей постановки вопроса.

Сейчас в стране, как снежный ком, растет количество бирж, брокерских контор, коммерческих банков, маленьких фирм, больших фирм, имеющих филиалы и отделения по всей стране. Все они заинтересованы в оперативном обмене информацией. Пока что у нас берегутся в основном государственные да военные секреты, тщательно прячутся партийные тайны, но большинство людей от только что народившегося отечественного бизнеса не особенно утруждают себя защитой коммерческой информации. Видимо, потребуется два–три громких скандала с миллиардными убытками, чтобы заинтересованные люди начали задумываться об этих проблемах.

Уже были эпизоды, когда, например, прибалтийские хакеры успешно ломали "защищенные" от них системы на АТС и в прочих подобных местах, затем успешно их чинили. Похоже, что на этих предприятиях даже не заметили постороннего вмешательства. Вряд ли кто–то сможет помешать им вскрыть банковскую или биржевую систему...

В СССР сложилась странная ситуация: все работают на персоналках, нормальным считается покупать самые дорогие компьютеры на 80486 и потом гонять их под MS–DOS (это с 8 Мбайтами ОЗУ!), причем такая конфигурация является просто нормой жизни для всех и вся (исключений крайне мало, несмотря на то, что существуют другие машины, другие операционные системы, специализированное программное обеспечение для них, которое во многих случаях может оказаться очень полезным). Происходит это в основном потому, что фирмы, поставляющие вычислительную технику в нашу страну не утруждают себя изучением нужд клиента – им это не нужно, важнее успеть привезти, подготовить к продаже и сбыть. Покупатель берет все, особенно то, что подешевле, так что нужно просто успевать за изменением его желаний. А они развиваются элементарно: 8088 –> 80286 –> 80386 –> 80486 –> ...

Таким поставщикам выгодно скрывать информацию о других (идеологически) системах технических и программных средствах, позволяющих создавать более эффективные, соответствующие интересам пользователя системы. В том числе и более безопасные. Это сложно, это дороже, требует бо́льших знаний и существенных предварительных затрат – короче, это не для тех, кому важна быстрая нажива при минимальных трудностях. Поэтому насаждается определенный стереотип восприятия компьютера, от которого пользователь боится отступить хотя бы на шаг, либо просто не имеет представления о других возможных конфигурациях нужной ему системы.

Первая наша статья рассказывает о принципиальной уязвимости операционной системы MS–DOS. Эти соображения можно распространить и на другие MS–DOS–совместимые системы. Важно понимать то, что это – однопользовательские операционные системы, и при их создании просто не стояла задача обеспечения защиты информации в системе и самой системы. Существует несколько популярных систем с достаточно проработанной подсистемой обеспечения безопасности. Наиболее часто применяются UNIX и VAX/VMS. Эти системы подходят для обеспечения работы как мощных центральных компьютеров, так и небольших настольных машин (обычно в

сети). Конечно, ресурсов требуется гораздо больше, чем при работе с MS-DOS, но и возможности качественно другие.

В цивилизованном мире вопросы безопасности рассматриваются наравне с вопросами эффективности и надежности системы. Ими серьезно занимаются многие солидные организации и фирмы. Существенно, что положение на рынке зависит от категории безопасности, которую обеспечивает поставляемая техника. Поэтому грамотные пользователи требуют от поставщиков создания систем на базе программного и аппаратного обеспечения, объективно способного обеспечить возможность безопасной работы. Производители, в свою очередь, стремятся добиться категорирования своих продуктов. Это играет существенную роль при получении крупных и выгодных заказов от банков, бирж, больших фирм, от обороны.

В нашей стране пока что эти вопросы наиболее занимают компетентные органы. В ряде таких организаций накоплен значительный научный и практический потенциал в данной области. Может быть имеет смысл подумать о том, чтобы в ходе конверсии использовать и эти знания? Ведь с развитием рынка, конкуренции, неизбежно будут появляться различные не слишком корректные методы борьбы с конкурентами... Сейчас те люди, которые могут помочь с консультациями в данной области, тихо отмалчиваются, стараясь не показываться честному люду. Может быть пора уже подумать о деле, хватит беречь секреты "ума, чести и совести". Существуют многие другие важные виды информации.

Только в Москве работает свыше 900 брокерских контор в которых трудятся десять с лишним тысяч брокеров. Многие из них связаны между собой совершенно беззащитными системами передачи информации. Излюбленный метод передачи документов с помощью факса – классический пример нашей безалаберности. Тем, кто использует для коммуникаций компьютеры, мы напоминаем, что существуют вирусы, предназначенные специально для вскрытия таких систем, причем устроены они так, что системе наносится серьезный ущерб. Как правило такие вирусы (или прочие вредительские программы) пишутся не развлекающимися хакерами, а серьезными специалистами, прекрасно осознающими, какой эффект должен быть получен в каждом конкретном случае и как его добиться с минимальной вероятностью противодействия. Здесь речь идет скорее о Западе, но, по мере развития бизнеса у нас, весьма вероятно энергичное развитие промышленного шпионажа. Тогда действительно может появиться целый клан специалистов, работающих в этой области. Кстати, военные и государственные тайны тоже должны оставаться тайнами, а эти люди могут, в принципе, расширить поле своей деятельности...

На Западе, на который мы постоянно киваем, вопросы безопасности начинают прорабатываться уже на этапе планирования разработки того или иного продукта. Так может быть есть смысл уже сегодня подумать о том, как нам работать завтра, и не начинать проектировать системы, которые однажды станут причиной многих бед из-за собственной беззащитности?

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

"Остап подошел к двери, сунул в щель американского замка длинный желтый ноготь большого пальца и осторожно стал поворачивать его справа налево и сверху вниз. Дверь бесшумно отворилась..."

"Двенадцать стульев" И.Ильф, Е.Петров

"Дыры" в MS-DOS и программы защиты информации

Введение

В предлагаемой статье приводятся результаты анализа "слабых мест" систем разграничения доступа (СРД) в операционной системе MS-DOS. Более или менее подробно рассматриваются способы снятия защиты программных компонент СРД и извлечения информации из подобных систем.

Открытая и хорошо документированная MS-DOS не позволяет системными методами вводить ограничения на пользование ресурсами компьютера (оперативной памятью, дисками, принтерами и так далее). И в результате у пользователей появляется потребность в продуктах, исправляющих этот недостаток DOS, а у производителей таких продуктов — сильная головная боль, вызываемая возникающими при этом проблемами. Легче всего (и чаще всего) реализуются программные СРД, основанные на подробном исследовании функций MS-DOS. К тому же они значительно дешевле аналогичных аппаратных систем. Однако многие подобные СРД не обеспечивают защиты, если "взломщик" имеет более детальные знания о структуре и свойствах MS-DOS. Приводимые в связи с этим рассуждения могут навести производителей и пользователей программных СРД на тревожные мысли в отношении уже разработанных и используемых систем, не обеспечивающих достаточной безопасности информации.

Для усиления защитных свойств программных СРД могу лишь порекомендовать: во-первых, постоянную верификацию компонентов системы в процессе ее работы, то есть проверку векторов прерываний, целостности некоторых таблиц и программ DOS, наиболее важных участков кодов и данных верифицируемой СРД; во-вторых, шифрование всей или части защищаемой информации. Хотя при некотором мыслительном усилии и в этих идеях можно найти массу "дыр". Поэтому единственный известный мне способ создания действительно надежной СРД — это обязательная аппаратная поддержка системы на любом ее уровне.

Кроме того, изложенное в статье сильно напоминает руководство по "взлому" программного обеспечения с изложением "дыр" MS-DOS, о которых лучше помалкивать.

По этому поводу хочу заметить, что в статье не описано ни одного нового, не известного специалистам, способа обхода СРД. Многие из этих методов уже давно обнаружены, испытаны, описаны в литературе (например, в электронных справочниках "Interrupt List" by Ralf Brown и "Tech Help" by Dan Rollins) и используются (например, некоторыми компьютерными вирусами). Но, к сожалению, не все производители программ разграничения доступа отдают должное внимание "дырам" MS-DOS, на что я и хочу обратить их внимание в этой статье.

Системы разграничения доступа

Большее число комплексов защиты от несанкционированного доступа на компьютерах типа IBM PC разработаны либо только в программном исполнении, либо, в некотором смысле, с минимальной реализацией аппаратной части, что не способно защитить подобные системы от проникновения квалифицированного "злоумышленника". Рассматриваемая в статье модель "компьютерная система — злоумышленник" соответствует следующим предположениям:

1) компьютерная система представляет из себя один или несколько (вычислительная сеть) компьютеров типа IBM PC, снабженных:
а) стандартной системой ввода/вывода BIOS (то есть не имеющей встроенных средств разграничения доступа) и операционной системой MS-DOS;
б) программным или программно-аппаратным комплексом разграничения доступа.

2) "злоумышленник" не знает ключей или паролей доступа, но имеет возможность кратковременного доступа к защищенной системе, то есть загрузки компьютера (компьютеров) системы с собственной дискеты или запуска собственной программы ("программа-шпион").

Известные мне СРД основаны на обработке обращений пользователей к файлам, к ресурсам операционной системы (оперативная память) и внешним устройствам (дискам, в том числе и удаленным, стримерам и так далее). При этом подобные СРД перехватывают соответствующие аппаратные и (или) программные прерывания системы, анализируют их и, в зависимости от уровня доступа пользователя, разрешают или запрещают исполнение этих прерываний. К таким прерываниям относятся прерывания с номерами 21h (вызов функций DOS), 13h (обращение к винчестеру или флоппи-диску) и 40h (обращение к флоппи-диску) и некоторые другие.

Программная реализация таких систем выполняется следующими методами:

1) встраивание в цепочку обработчика прерывания (рис.1);

2) встраивание в DOS (рис.2).

При встраивании в цепочку прерывания СРД определяет и запоминает значение вектора прерывания, а затем устанавливает новое значение вектора на собственную подпрограмму обработки. При вызове прерывания СРД перехватывает его, обрабатывает и, если пользователю разрешен вызов этого прерывания, передает дальше по цепочке, используя сохраненный адрес.

При встраивании в DOS СРД действует практически тем же способом, но только встраивается в самый конец цепочки, непосредственно перед DOS. Для

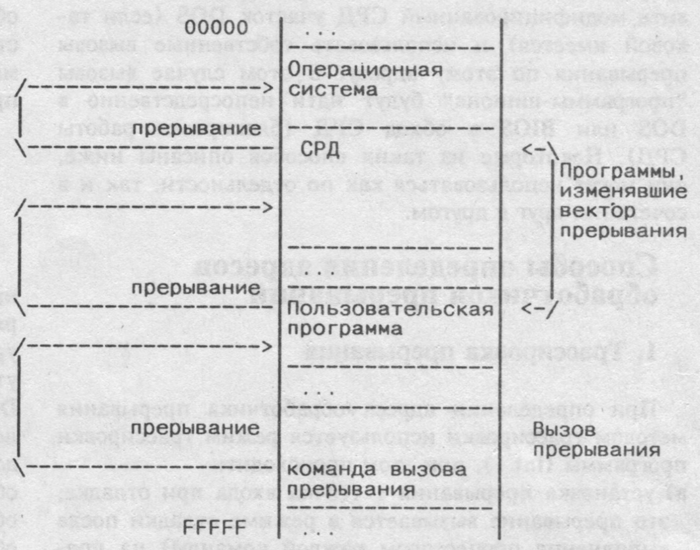


Рис.1 Встраивание в цепочку прерывания.

этого СРД помещает в нужное место внутри DOS команды перехода на собственную подпрограмму обработчика прерывания. Естественно, что при этом СРД несколько модифицирует DOS и должна принимать меры, которые гарантируют правильную работу модифицированной DOS.

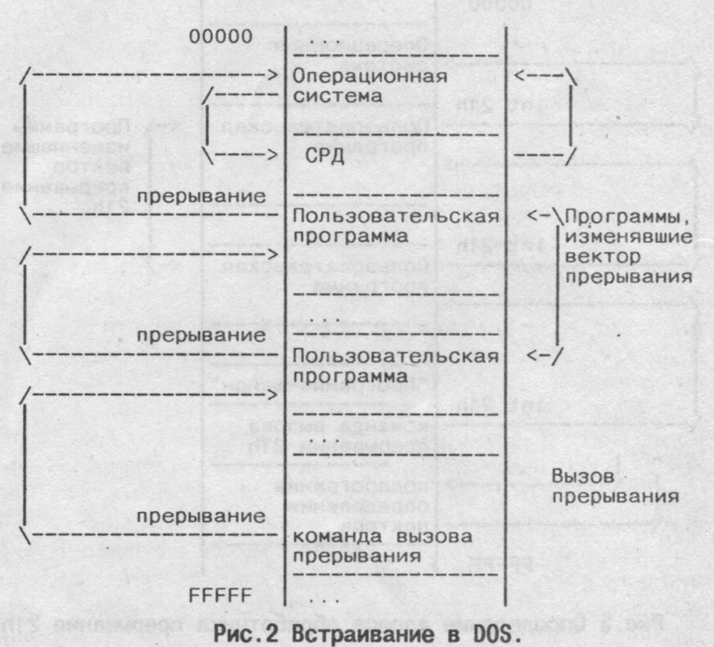


Рис.2 Встраивание в DOS.

Возможны вариации описанных методов и совместное их использование (проверка "сверху" и "снизу").

Для того чтобы обойти подобную защиту, злоумышленник должен блокировать работу СРД, то есть определить адрес обработчика прерывания в DOS (для прерываний 13h и 40h — в DOS или BIOS), исправить модифицированный СРД участок DOS (если таковой имеется) и использовать собственные вызовы прерывания по этому адресу. В этом случае вызовы "программы-шпиона" будут идти непосредственно в DOS или BIOS в обход СРД (блокировка работы СРД). Некоторые из таких способов описаны ниже, они могут использоваться как по отдельности, так и в сочетании друг с другом.

Способы определения адресов обработчиков прерываний

1. Трассировка прерывания

При определении адреса обработчика прерывания методом трассировки используется режим трассировки программы (int 1), при этом происходит:

а) установка прерывания 1 (точка входа при отладке; это прерывание вызывается в режиме отладки после выполнения процессором каждой команды) на подпрограмму определения истинного значения вектора прерывания;

б) включение режима отладки;

в) вызов в некотором смысле "безопасной" функции анализируемого прерывания.

При выполнении "безопасной" функции прерывания последовательно исполняются команды всех программ, которые изменяли вектор этого прерывания, затем выполняются команды DOS или BIOS, обрабатывающие данное прерывание. При этом после каждой выполненной команды управление получает "программа-шпион", определяющая адрес обработчика прерывания. Она анализирует адрес последней выполненной команды и, если этот адрес указывает на область памяти, принадлежащую операционной системе, выключает режим отладки, а адрес данной команды рассматривает как найденное значение вектора прерывания (рис.3).

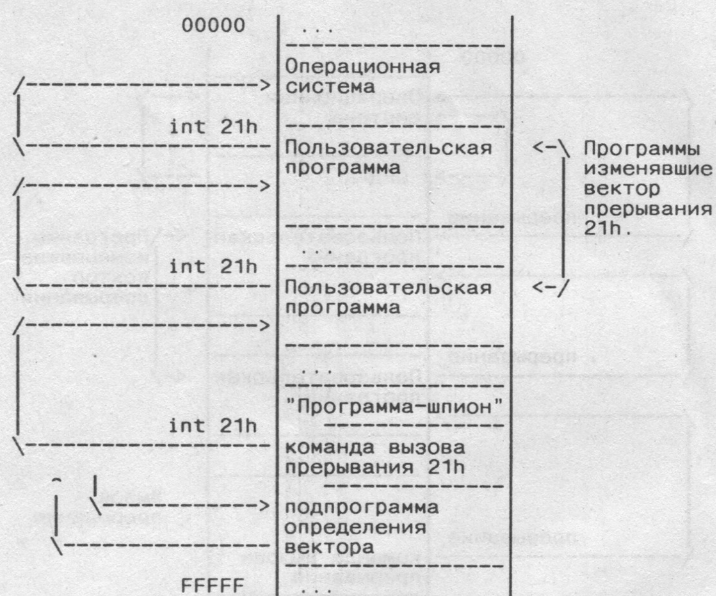


Рис.3 Определение адреса обработчика прерывания 21h.

2. Сканирование памяти и метод фиксированных адресов

Большинство версий MS-DOS и BIOS, используемых на IBM PC, в большей или меньшей степени повторяют друг друга, причем не только на логическом уровне, но и на уровне выполняемых кодов. Это утверждение верно для обработчика прерывания 21h в DOS и прерывания 13h в BIOS. Поэтому для определения адреса обработчика прерывания часто бывает достаточно просканировать сегмент, содержащий коды обработчика, на наличие команд, которые могут этому обработчику принадлежать. Адрес сегмента BIOS обычно равен C800h, F000h или F800h, на этот сегмент указывают вектора многих аппаратных прерываний. Адрес сегмента DOS, содержащего обработчик прерывания 21h, можно определить несколькими способами (приведенный список можно продолжить):

а) вектора практически всех прерываний DOS (int 20h — 3Fh) при ее инициализации указывают на программы, расположенные в сегменте DOS. В дальнейшем эти вектора изменяются, но некоторые из них (особенно редко использующиеся с номерами 34h — 3Fh) продолжают указывать на сегмент DOS;

б) точка входа CP/M также расположена в сегменте DOS;

в) недокументированная функция 34h прерывания 21h возвращает в регистрах ES:BX адрес флага активности DOS, который расположен в сегменте DOS;

г) недокументированная функция 52h прерывания 21h возвращает в регистрах ES:BX адрес списка списков (List of Lists), который расположен в сегменте DOS;

д) подфункция 3 функции 12h прерывания 2Fh возвращает в регистре DS адрес сегмента DOS.

Для конкретных версий DOS сканирование памяти можно провести по фиксированным адресам, которые являются точками входа в DOS для прерываний 13h и 21h, или использовать значения системных переменных DOS, содержащих вектор прерывания 13h в BIOS. К тому же адрес входа прерывания 21h не меняется в различных реализациях наиболее популярной версии DOS 3.x, а адрес участка,

содержащего значение вектора прерывания 13h, не изменился и при переходе к DOS 4.0.

Листинги обработчиков прерываний 13h и 21h в DOS 3.30, установленной на компьютере PC/AT (BIOS фирмы ARC), и DOS 4.01 на компьютере PS/2 модели 55SX (BIOS фирмы IBM) приведены в Приложении.

3. Использование прерывания 2Fh для определения адреса прерывания 13h

Функция 13h прерывания 2Fh используется внутри DOS для определения и одновременной установки внутрисистемных адресов, по которым DOS обращается в прерыванию 13h. При вызове этой функции регистрам DS:DX необходимо присвоить адрес, на который будет указывать вектор прерывания 13h после вызова функции. В тех же регистрах будет возвращен адрес обработчика прерывания 13h в BIOS.

Пример использования:

```
MOV  AX,1300h
INT  2Fh  ;Взять в DS:BX вектор прерывания 13h.
PUSH DS   ;Сохранить полученное значение.
PUSH BX
INT  2Fh  ;Восстановить вектор прерывания 13h.
POP  BX   ;Взять в DS:BX значение
POP  DS   ;вектора прерывания 13h.
```

Использование “задних дверей“ при вызове int 21h

Мне известны две такие “задние двери“ (back doors): вход CP/M и непрямой вызов DOS — функция 5Dh.

Для совместимости с операционной системой CP/M оставлена “задняя дверь“ входа в обработчик прерывания 21h. Используя этот вход, можно обойти СРД, контролирующие прерывание 21h. Структура входа CP/M достаточно проста (см. Приложение): при запуске любой пользовательской программы DOS создает префикс ее программного сегмента (PSP). По адресу PSP_SEG:0005 лежит команда длинного вызова подпрограммы (CALL FAR). Адрес, куда передается управление, указывает на таблицу векторов прерываний (адрес 0000:00C0). По этому адресу находится 5 байт (они перекрывают вектора int 30h и int 31h) команды длинного перехода (JMP FAR) на обработчик входа CP/M, который расположен в DOS непосредственно перед обработчиком прерывания 21h. Способы использования этого входа “программой-шпионом“ достаточно очевидны.

Косвенный вызов MS-DOS реализован во многих ее версиях как внутренняя функция. При ее использовании необходимо поместить в регистры DS:DX указатель на массив значений системных регистров, в регистр AX занести значение 5D00h и вызвать прерывание 21h. При выполнении функции 5D00h DOS заполнит регистры (в том числе и регистр AX) из массива значений и выполнит указанную в регистре AX функцию. Вызов функции в данном случае произойдет через дополнительную точку входа, расположенную значительно ниже “официального“ входа в прерывание 21h.

Чтение по абсолютному адресу и изменение прав доступа к файлам

Если СРД запрещает доступ к закрытым файлам, блокируя только прерывание 21h (или если “программа-шпион“ уже сняла защиту с прерывания 13h), то возможно посекторное чтение из файлов или запись в них с использованием:

а) обращения к драйверу диска;
б) прерываний 13h или 25h/26h.

При вызове этих прерываний или при организации запроса драйвера необходимо определить местоположение файла на диске, то есть адреса секторов, принадлежащих файлу (логические номера секторов для драйвера или прерываний 25h/26h и абсолютные — трек/головка/сектор — для прерывания 13h). Адрес первого сектора файла достаточно просто вычислить, зная номер первого кластера файла. Номер первого кластера определяется несколькими способами. Самый надежный, хотя и не самый простой, способ заключается в последовательном просмотре секторов, содержащих подкаталоги, входящие в полное имя файла. В определенных полях этих секторов находятся адреса начальных кластеров /элементов каталога (файлов и подкаталогов). Номер начального кластера файла можно найти также в некоторых недокументированных таблицах DOS (например, System File Table) или в определенных полях FCB после использования функций FindFirst и FindNext.

Если пользователю разрешено только чтение из файла, то определенными действиями он может разрешить запись в файл и даже уничтожение файла. Эти можно проделать довольно незаметно для DOS и СРД, изменив права доступа к файлу. Для этого достаточно несколько модифицировать некоторые недокументированные области DOS, например системную таблицу файла (System File Table).

Е.Касперский
телефон (095)499-15-00

ПРИЛОЖЕНИЕ

ЛИСТИНГ ОБЛАСТЕЙ DOS и BIOS (IBM PS/2, DOS 4.01, BIOS фирмы IBM)

Таблица векторов прерываний (сегмент 0000h)

```
0000:004C  21F8 0070   INT_13h_Vector  DD    007021F8h                 ; Вектор прерывания
                                                                       ; 13h
0000:00C0  EA 02BE:16F1                JMP   FAR PTR CP/M_Entry        ; (02BE:16F1)
                                       . . .                           ; прерывания 21h
```

Область IO.SYS (сегмент 0070h)

```
0070:00B0  53EC F000   INT_13h_BIOS_2  DD    0F00053ECh                ; Адрес ???
0070:00B4  53EC F000   INT_13h_BIOS    DD    0F00053ECh                ; Адрес прерывания
                                       . . .                           ; 13h
0070:0248  1E93 0070   INT_2Fh_Redirct DD    00701E93h                 ; Адрес прерывания
2Fh
                                       . . .
          ======================================================================
          | Подпрограмма вызова прерывания 13h в BIOS,
          |   вызывается из обработчика прерывания 13h в DOS
          |   (INT_13h_DOS, адрес 0070:21F8)
          ======================================================================
                       INT_13h_Caller  PROC  FAR
0070:1216  2E: A3 00C0                 MOV   CS:[00C0],AX              ; Запомнить AX
0070:121A  9C                          PUSHF                           ; Сохранить флаги
0070:121B  80 FC 05                    CMP   AH,5                      ; Функция 5?
0070:121E  75 0A                       JNE   Not_func_5                ; (format)
0070:1220  2E: C7 06 0252 0140         MOV   CS:[0252],140h            ; Обработка
0070:1227  E8 1006                     CALL  2230                      ; функции 5
0070:122A              Not_func_5:
0070:122A  80 FC 08                    CMP   AH,8                      ; Функция 8?
0070:122D  74 12                       JE    Func_8
0070:122F  80 FC 15                    CMP   AH,15h                    ; Функция 15h?
0070:1232  74 0D                       JE    Func_15
0070:1234  2E: FF 1E 00B4              CALL  CS:INT_13h_BIOS           ; (0070:00B4)
                                                                       ; Вызов int 13h в
0070:1239  72 03                       JC    Error                     ; BIOS
0070:123B  CA 0002                     RETF  2
          ======================================================================
          | Обработчик прерывания 2Fh
          ======================================================================
                       INT_2Fh         PROC  FAR
0070:1C74  80 FC 13                    CMP   AH,13h                    ; Функция 13h?
0070:1C77  74 05                       JE    Int_2F_f_13
0070:1C79  2E: FF 26 0248              JMP   WORD PTR CS:INT_2Fh_Redirct ; (0070:0248)
                                                                       ; Передача
0070:1C7E              Int_2F_f_13:                                    ; управления, если
0070:1C7E  2E: FF 36 00B4              PUSH  WORD PTR CS:INT_13h_BIOS  ; нет (0070:00B4)
0070:1C83  2E: FF 36 00B6              PUSH  WORD PTR CS:INT_13h_BIOS+2 ; (0070:00B6)
0070:1C88  2E: FF 36 00B0              PUSH  WORD PTR CS:INT_13h_BIOS_2 ; (0070:00B0)
0070:1C8D  2E: FF 36 00B2              PUSH  WORD PTR CS:INT_13h_BIOS_2+2 ; (0070:00B2)
0070:1C92  2E: 89 16 00B4              MOV   WORD PTR CS:INT_13h_BIOS,DX ; (0070:00B4)
0070:1C97  2E: 8C 1E 00B6              MOV   WORD PTR CS:INT_13h_BIOS+2,DS ; (0070:00B6)
0070:1C9C  2E: 89 1E 00B0              MOV   WORD PTR CS:INT_13h_BIOS_2,BX ; (0070:00B0)
0070:1CA1  2E: 8C 06 00B2              MOV   WORD PTR CS:INT_13h_BIOS_2+2,ES ; (0070:00B2)
0070:1CA6  07                          POP   ES                        ; Установка и
0070:1CA7  5B                          POP   BX                        ; возврат адреса
0070:1CA8  1F                          POP   DS                        ; int 13h в BIOS
0070:1CA9  5A                          POP   DX
0070:1CAA  CF                          IRET
                       INT_2Fh         ENDP
                                       . . .
0070:21EE  1216 0070   INT_13h_Addr    DD    00701216h                 ; Адрес п/п вызова
                                                                       ; int 13h
0070:21F2  11BA 0070   INT_13h_CS_IP   DD    007011BAh                 ; CS:IP при вызове
                                                                       ; int 13h
0070:21F6  7246        INT_13h_Flags   DW    7246h                     ; Флаги при вызове
                                                                       ; int 13h
```

```
                 ============================================================
                 | Обработчик прерывания 13h в DOS
                 ============================================================

                         INT_13h_DOS       PROC   FAR
0070:21F8  2E: 8F 06 21F2                  POP    WORD PTR CS:INT_13h_CS_IP      ; Запомнить CS
0070:21FD  2E: 8F 06 21F4                  POP    WORD PTR CS:INT_13h_CS_IP+2    ; Запомнить IP
0070:2202  2E: 8F 06 21F6                  POP    CS:INT_13h_Flags               ; Запомнить флаги
0070:2207  9C                              PUSHF
0070:2208  2E: FF 1E 21EE                  CALL   CS:INT_13h_Addr                ; Вызов п/п
                                                                                 ; INT_13h_Caller
0070:220D  72 05                           JC     Error
0070:220F  2E: FF 2E 21F2                  JMP    CS:INT_13h_CS_IP               ; Возврат из int 13h
                                           . . .

                 Область MSDOS.SYS (сегмент 02BEh)

02BE:0584  0E08          CP/M_IP           DW     0E08h                          ; IP при вызове CP/M
                                           . . .
02BE:0EBB  24            CP/M_MAX_Func_N   DB     24h                            ; Максимальный N
                                                                                 ; функции CP/M
02BE:0EBC  6C            DOS_MAX_Func_N    DB     6Ch                            ; Максимальный N
                                                                                 ; функции DOS
                                           . . .
02BE:16EE                Error_No:
02BE:16EE  B0 00                           MOV    AL,0
02BE:16F0  CF                              IRET

                 ============================================================
                 | Обработчик CP/M
                 ============================================================

02BE:16F1                CP/M_Entry:
02BE:16F1  58                              POP    AX                             ; Подготовка стека
02BE:16F2  58                              POP    AX                             ; для входа
02BE:16F3  2E: 8F 06 0584                  POP    CS:CP/M_IP                     ; в int 21h
02BE:16F8  9C                              PUSHF
02BE:16F9  FA                              CLI
02BE:16FA  50                              PUSH   AX
02BE:16FB  2E: FF 36 0584                  PUSH   CS:CP/M_IP
02BE:1700  2E: 3A 0E 0EBB                  CMP    CL,CS:CP/M_MAX_Func_N          ; Допустимая
02BE:1705  77 E7                           JA     Error_No                       ; функция?
02BE:1707  8A E1                           MOV    AH,CL
02BE:1709  EB 07                           JMP    SHORT Ext_Entry                ; Переход в int 21h

                 ============================================================
                 | Обработчик прерывания 21h
                 ============================================================

                         INT_21h           PROC   FAR
02BE:170B  2E: 3A 26 0EBC                  CMP    AH,CS:DOS_MAX_Func_N           ; Допустимая
02BE:1710  77 DC                           JA     Error_No                       ; функция?
02BE:1712                Ext_Entry:                                              ; Точка входа из
                                                                                 ; CP/M
02BE:1712  80 FC 51                        CMP    AH,51h                         ; Обработка вызова
02BE:1715  74 C8                           JE     Func_51                        ; int 21h
02BE:1717  80 FC 62                        CMP    AH,62h
02BE:171A  74 C3                           JE     Func_62
                                           . . .

                 Область PSP выполняемых программ (сегмент Segm)

                                           . . .
Segm:0005  9A F01D:FEF0                    CALL   FAR PTR CP/M                   ; Вызов CP/M
                                           . . .

                         Область BIOS (сегмент F000h)

                 ============================================================
                 | Обработчик прерывания 13h в BIOS
                 ============================================================

                         INT_13h_BIOS      PROC   FAR
F000:53EC  80 FA 80                        CMP    DL,80h                         ; Флоппи или
                                                                                 ; винчестер?
F000:53EF  72 3C                           JB     Floppy                         ; Переход на флоппи
F000:53F1  FB                              STI
```

```
F000:53F2   0A E4                          OR       AH,AH                          ; Сброс?
F000:53F4   75 04                          JNZ      Not_func_0
F000:53F6   CD 40                          INT      40h                            ; Сброс флоппи
F000:53F8   2A E4                          SUB      AH,AH
F000:53FA                Not_func_0:
                                           . . .                                   ;  для винчестера
F000:542D                Floppy:
F000:542D   CD 40                          INT      40h                            ; Вызов обработчика
F000:542F   CA 0002                        RETF     2                              ; флоппи
                                           . . .
```

ЛИСТИНГ ОБЛАСТЕЙ DOS и BIOS (IBM PC/AT, DOS 3.30, BIOS фирмы ARC)

Таблица векторов прерываний (сегмент 0000h)

```
0000:004C   1DB7 0070    INT_13h_Vector    DD       00701DB7h                      ; Вектор прерывания
13h
                                           . . .
0000:00C0   EA 0283:1446                   JMP      FAR PTR CP/M_Entry             ; (0283:1446)
                                           . . .                                   ; Переход на точку
CP/M
```

Область IO.SYS (сегмент 0070h)

```
0070:00B0   A069 F000    INT_13h_BIOS_2    DD       0F000A069h                     ; Адрес ???
0070:00B4   A069 F000    INT_13h_BIOS      DD       0F000A069h                     ; Адрес прерывания
13h
                                           . . .
0070:0248   1ABF 0070    INT_2Fh_Redirct   DD       00701ABFh                      ; Адрес прерывания
2Fh
                   ============================================================
                   | Подпрограмма вызова прерывания 13h в BIOS,
                   |   вызывается из обработчика прерывания 13h в DOS
                   |   (INT_13h_DOS, адрес 0070:1DB7)
                   ============================================================
                         INT_13h_Caller    PROC     FAR
0070:0F9D   2E: A3 00C0                    MOV      CS:[00C0],AX                   ; Запомнить AX
0070:0FA1   9C                             PUSHF                                   ; Сохранить флаги
0070:0FA2   80 FC 05                       CMP      AH,5                           ; Функция 5?
(format)
0070:0FA5   75 0A                          JNE      Not_func_5
0070:0FA7   2E: C7 06 0252 0140            MOV      CS:[0252],140h                 ; Обработка функции
5
0070:0FAE   E8 0E3E                        CALL     1DEF
0070:0FB1                Not_func_5:
0070:0FB1   80 FC 08                       CMP      AH,8                           ; Функция 8?
0070:0FB4   74 12                          JE       Func_8
0070:0FB6   80 FC 15                       CMP      AH,15h                         ; Функция 15h?
0070:0FB9   74 0D                          JE       Func_15
0070:0FBB   2E: FF 1E 00B4                 CALL     CS:INT_13h_BIOS                ; (0070:00B4)
                                                                                   ; Вызов int 13h в
BIOS
0070:0FC0   72 03                          JC       Error
0070:0FC2   CA 0002                        RETF     2
                                           . . .
                   ============================================================
                   | Обработчик прерывания 2Fh
                   ============================================================

                         INT_2Fh           PROC     FAR
0070:1853   80 FC 13                       CMP      AH,13h                         ; Функция 13h?
0070:1856   74 05                          JE       Int_2F_f_13
0070:1858   2E: FF 26 0248                 JMP      WORD PTR CS:INT_2Fh_Redirct    ; (0070:0248)
                                                                                   ; Передача
управления,
0070:185D                Int_2F_f_13:                                              ;  если нет
0070:185D   2E: FF 36 00B4                 PUSH     WORD PTR CS:INT_13h_BIOS       ; (0070:00B4)
0070:1862   2E: FF 36 00B6                 PUSH     WORD PTR CS:INT_13h_BIOS+2     ; (0070:00B6)
0070:1867   2E: FF 36 00B0                 PUSH     WORD PTR CS:INT_13h_BIOS_2     ; (0070:00B0)
0070:186C   2E: FF 36 00B2                 PUSH     WORD PTR CS:INT_13h_BIOS_2+2   ; (0070:00B2)
0070:1871   2E: 89 16 00B4                 MOV      WORD PTR CS:INT_13h_BIOS,DX    ; (0070:00B4)
0070:1876   2E: 8C 1E 00B6                 MOV      WORD PTR CS:INT_13h_BIOS+2,DS  ; (0070:00B6)
0070:187B   2E: 89 1E 00B0                 MOV      WORD PTR CS:INT_13h_BIOS_2,BX  ; (0070:00B0)
```

```
0070:1880  2E: 8C 06 00B2                MOV    WORD PTR CS:INT_13h_BIOS_2+2,ES  ; (0070:00B2)
0070:1885  07                            POP    ES                               ; Установка и
                                                                                 ; возврат
0070:1886  5B                            POP    BX                               ; адреса int 13h
                                                                                 ; в BIOS
0070:1887  1F                            POP    DS
0070:1888  5A                            POP    DX
0070:1889  CF                            IRET
                      INT_2Fh            ENDP
                                         . . .
0070:1DAD  0F9D 0070  INT_13h_Addr       DD     00700F9Dh                        ; Адрес п/п
                                                                                 ; вызова int 13h
0070:1DB1  0F41 0070  INT_13h_CS_IP      DD     00700F41h                        ; CS:IP при вызове
                                                                                 ; int 13h
0070:1DB5  0246       INT_13h_Flags      DW     0246h                            ; Флаги при вызове
                                                                                 ; int 13h

                ====================================================================
                | Обработчик прерывания 13h в DOS
                ====================================================================

                      INT_13h_DOS        PROC   FAR
0070:1DB7  2E: 8F 06 1DB1                POP    WORD PTR CS:INT_13h_CS_IP        ; Запомнить CS
0070:1DBC  2E: 8F 06 1DB3                POP    WORD PTR CS:INT_13h_CS_IP+2      ; Запомнить IP
0070:1DC1  2E: 8F 06 1DB5                POP    CS:INT_13h_Flags                 ; Запомнить флаги
0070:1DC6  9C                            PUSHF
0070:1DC7  2E: FF 1E 1DAD                CALL   CS:INT_13h_Addr                  ; Вызов п/п
                                                                                 ; INT_13h_Caller
0070:1DCC  72 05                         JC     Error
0070:1DCE  2E: FF 2E 1DB1                JMP    CS:INT_13h_CS_IP                 ; Возврат из int 13h
                                         . . .

                 Область MSDOS.SYS (сегмент 0283h)

0283:051E  0825       CP/M_IP            DW     0825h                            ; IP при вызове CP/M
                                         . . .
0283:0DFE  24         CP/M_MAX_Func_N    DB     24h                              ; Максимальный N
функции CP/M
0283:0DFF  68         DOS_MAX_Func_N     DB     68h                              ; Максимальный N
функции DOS
                                         . . .
0283:1443             Error_No:
0283:1443  B0 00                         MOV    AL,0
0283:1445  CF                            IRET

                ====================================================================
                | Обработчик CP/M
                ====================================================================
0283:1446             CP/M_Entry:
0283:1446  58                            POP    AX                               ; Подготовка стека
0283:1447  58                            POP    AX                               ; для входа
0283:1448  2E: 8F 06 051E                POP    CS:CP/M_IP                       ; в int 21h
0283:144D  9C                            PUSHF
0283:144E  FA                            CLI
0283:144F  50                            PUSH   AX
0283:1450  2E: FF 36 051E                PUSH   CS:CP/M_IP
0283:1455  2E: 3A 0E 0DFE                CMP    CL,CS:CP/M_MAX_Func_N            ; Допустимая
                                                                                 ; функция?
0283:145A  77 E7                         JA     Error_No
0283:145C  8A E1                         MOV    AH,CL
0283:145E  EB 07                         JMP    SHORT Ext_Entry                  ; Переход в int 21h

                ====================================================================
                | Обработчик прерывания 21h
                ====================================================================

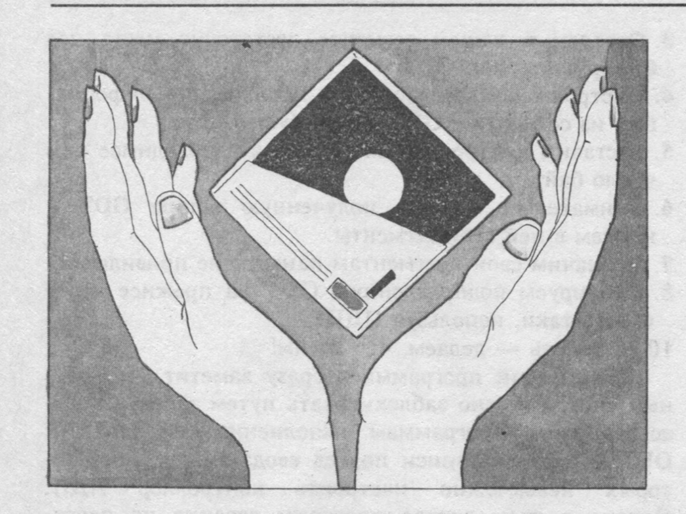
                      INT_21h            PROC   FAR
0283:1460  2E: 3A 26 0DFF                CMP    AH,CS:DOS_MAX_Func_N             ; Допустимая
                                                                                 ; функция?
0283:1465  77 DC                         JA     Error_No
0283:1467             Ext_Entry:                                                 ; Точка входа
                                                                                 ; из CP/M
0283:1467  80 FC 51                      CMP    AH,51h                           ; Обработка вызова
0283:146A  74 A1                         JE     Func_51                          ; int 21h
0283:146C  80 FC 62                      CMP    AH,62h
0283:146F  74 9C                         JE     Func_62
```

```
        Область PSP выполняемых программ (сегмент Segm)
                                        . . .
Segm:0005  9A F01D:FEF0                 CALL    FAR PTR CP/M              ; Вызов CP/M
                                        . . .

             Область BIOS (сегмент F000h)

          =========================================================
          | Обработчик прерывания 13h в BIOS
          =========================================================

                         INT_13h_BIOS   PROC    FAR
F000:A069  80 FA 80                     CMP     DL,80h                    ; Флоппи или
                                                                          ; винчестер?
F000:A06C  FB                           STI
F000:A06D  FC                           CLD
F000:A06E  72 53                        JC      Floppy                    ; Переход на флоппи
F000:A070  0A E4                        OR      AH,AH                     ; Сброс?
F000:A072  75 04                        JNZ     Not_func_0
F000:A074  CD 40                        INT     40h                       ; Сброс флоппи
F000:A076  32 E4                        XOR     AH,AH
F000:A078                Not_func_0:                                      ; Обработка вызова
                                        . . .                             ; int 13h для
                                                                          ; винчестера
F000:A0C3                Floppy:
F000:A0C3  CD 40                        INT     40h                       ; Вызов обработчика
F000:A0C5  CA 0002                      RETF    2                         ; флоппи
                                        . . .
```



В *последнее время в различных журналах появилось огромное число статей, посвященных двум самым больным вопросам советских программистов, — защите программ от вирусов и от несанкционированного копирования. Такой интерес вполне естествен: раз есть проблема, значит надо ее решать. Однако очень часто встречаются случаи, когда авторы утверждают, что они создали нечто такое, что позволит полностью защитить компьютер от заразы, а разработчика программ от грабежа...*

Некоторые соображения о защите программ

И действительно, существуют программы с очень надежной защитой от копирования (здесь в ход пускаются такие приемы, как кодирование программ при помощи сложных алгоритмов, хитроумные ловушки для трассировщиков, специальные микросхемы ПЗУ, заменяющие стандартный BIOS, различные "черные ящики", подключаемые к портам или вмонтированные в разъемы принтерных кабелей, ключевые дискеты и многое другое) или с хорошей, подчас изощренной, блокировкой действия вирусов. Нередко такие программы бывают удачными и хорошо работают (я и сам ими пользуюсь), но...

Именно это "но" перечеркивает все труды, вложенные в разработку систем защиты. Абсолютной защиты не существует! (Вспомните Рея Бредбери.) И в доказательство этого я хочу привести пару сюжетов с размышлениями о нашей бренной компьютерной жизни.

СЮЖЕТ ПЕРВЫЙ.

Прорыв антивирусной обороны

Большинство антивирусных систем работают стандартным образом. Они или проверяют выполняемые программы на наличие заражения по каким-либо признакам (прямым — присутствие в программе, на диске или в памяти в определенном месте заранее известной кодовой цепочки; косвенным — изменение длин файлов, контрольных сумм и др.), или, постоянно присутствуя в памяти машины (в виде резидентных программ или драйверов), контролируют жизненно важные точки операционной системы. Однако все подобные системы защиты от вирусов являются рудиментарными, вследствие того что MS DOS (а пока речь идет именно об этой операционной системе) сама по себе полностью беззащитна. Поэтому здесь можно говорить лишь о попытках залатать те или иные дыры, но ни коим образом не о глубоко эшелонированной и хорошо продуманной обороне. Положение резко изменится, если начать использовать надстройки DOS, работающие в защищенном режиме (protected mode). Эти программы так распределяют уровни привилегий при запуске пользовательских программ, что если даже в последних и содержится вирус, то он все равно не сможет поразить операционную систему. Любая злодейская попытка будет тут же пресечена при помощи механизма обработки критических ситуаций (процессоры i286 и старше расценивают попытку "залезть" в запрещенный сегмент или выполнить запретную команду как критическую ситуацию и вызы-

вают соответствующий обработчик). Теперь ликуй пользователь, ты надежно защищен! Но... Опять то самое проклятое "но".

Прежде чем продолжить, я должен объяснить, что привожу дальнейшие рассуждения не в угоду творцам вирусов. Адресованы они тем, кто безрассудно верует в вирусную устойчивость некоторых программных продуктов и поэтому может вовремя не принять надлежащих мер защиты. А что касается "технокрыс", то наверняка уже существуют вирусы, использующие подобные методы прорыва.

Система защиты процессора основана на том, что каждому сегменту памяти присваивается свой уровень привилегий. Если программа, расположенная в менее привилегированном кодовом сегменте, попытается "залезть" в более привилегированный сегмент, то процессор, который легко это обнаруживает, формирует критическую ситуацию. При этом контроль соответствия уровней привилегий осуществляется полностью автоматически. Информация об уровнях привилегий сегментов содержится в глобальной дескрипторной таблице (GDT), адрес которой хранится в регистре GDT. Так как на этой информации зиждется вся система защиты процессора, то имеется возможность запретить низкопривилегированным программам изменять содержимое регистра GDT, а путем присвоения сегменту с GDT наивысшей привилегии можно перекрыть возможность ее несанкционированного изменения. Именно так и поступают операционные системы, работающие в защищенном режиме. Своим сегментам они присваивают наивысший уровень привилегий, а всем порожденным процессам — более низкий. Казалось бы, надежнейшая защита, но она имеет тот недостаток, что компьютер здесь не рассматривается как система в целом.

Как же прорвать такую защиту? Ясно, что полностью ее обезвредить можно только путем изменения уровней привилегий в GDT. И сделать это оказывается очень просто. Достаточно использовать такой метод доступа к памяти, который не контролируется процессором, и имя этому методу — ПДП (прямой доступ к памяти). Действительно, практически во всех системах компьютеров имеются контроллеры ПДП, без которых немыслима мало-мальски терпимая работа с жесткими и гибкими дисками, стримерами, сканерами и другими устройствами. Именно эти контроллеры и можно использовать для взлома системы безопасности.

Итак, метод прорыва колец защиты виртуального режима (представим себя вирусом, пытающимся заразить компьютер):

1. Проверим, в каком режиме нас запустили. Если в реальном (обычном), то никаких хлопот, а если в виртуальном, то придется попотеть.
2. Прочитаем адрес глобальной дескрипторной таблицы (регистр GDT), благо, что разработчики i286 не предусмотрели возможности запрета операции чтения регистра GDT (LGDT), как они это сделали с операцией записи в регистр GDT (SGDT).
3. Отведем в нашем сегменте достаточно места под буфер для копии GDT.
4. Настроим контроллер ПДП на передачу информации из сегмента с GDT в наш буфер.
5. Заставим контроллер скопировать необходимое нам число байт.
6. Внимательно изучим полученную копию GDT и найдем в ней свои сегменты.
7. Назначим своим сегментам наивысшие привилегии.
8. Скопируем подправленную GDT на прежнее место (опять-таки, используя ПДП).
10. А теперь — делаем, что хотим!

Искушенный программист сразу заметит, что данный способ можно заблокировать путем запрета пользовательским программам выполнения команд IN и OUT (чтения и записи портов ввода-вывода), без которых невозможно настроить контроллер ПДП. Однако в этом случае программирование на персоналке потеряет всю свою прелесть и превратится в монотонную и рутинную работу, ибо программист не сможет даже запрограммировать адаптер EGA/VGA (а это означает, что придется распроститься даже с русификацией дисплея). Другим способом противодействия может стать регулярное сравнение операционной системой текущей GDT с ее резервной копией. При наличии рассогласования необходимо производить расследование его причин. Но нельзя слишком часто выполнять такое сравнение, так как это приведет к значительной потере производительности. Это значит, что все равно всегда найдется временное окно для вирусной атаки. И главный вывод здесь заключается в том, что, хоть на самое короткое мгновение, вопреки всем стражам все-таки можно сломать систему колец защиты процессора i286.

Каков же способ борьбы с напастями? Я не побоюсь показаться банальным и повторю старую истину: не пользуйтесь ворованными программами!

К сожалению, этот простой совет практически не выполним в наших условиях. Не каждый может выкроить свободных долларов этак 200 для покупки необходимого пакета, а то, что продается за рубли, либо слишком дорого для рядового пользователя, либо несколько не то, что ему нужно. Вот так и складывается знакомая до боли ситуация, когда позволить себе хоть что-нибудь купить могут только богатые предприятия. Руководители же этих предприятий в большинстве своем считают, что покупать нужно все, что угодно, только не программы. Зачем покупать, если можно просто списать?!

Выход у нас только один — постепенное формирование рынка программного обеспечения. Только при наличии такого рынка (и при его поддержке полноценной законодательной защитой авторских прав) могут увеличиться тиражи программ, а следовательно, упадут цены. Но пока еще рынок только формируется, а то, что его заменяет, дико по творимому беззаконию, и поэтому надо защищаться от воров самим. "Спасение утопающих — дело рук самих утопающих...", и перед разработчиком со всей

остротой встает проблема защиты программ от несанкционированного копирования и распространения.

СЮЖЕТ ВТОРОЙ. Как украсть миллион

Большинство советских производителей программного обеспечения пытаются в той или иной степени ограничить нелегальное копирование своих продуктов. Диапазон применяемых систем защиты очень велик — от простейших самопальных, которые часто можно снять путем изменения 1-2 байтов (например, достаточно бывает сменить условный переход на безусловный или наоборот), до высокопрофессиональных продуктов, созданных порой даже при участии специалистов-криптологов (есть случаи, когда методы кодирования, использованные в системах защиты, являлись составной частью диссертаций).

Но... эти системы защиты имеют несколько уязвимых мест, которые позволяют обойти все ухищрения их создателей. Первым уязвимым местом является момент принятия решения о легальности запущенной копии. Если вы с помощью отладчика добрались до этого места, то защита побеждена. Один мой знакомый как-то снимал зарубежную фирменную защиту (простим его за этот поступок), которая строилась на применении "черного ящика", подключаемого к параллельному порту. Нужного "ящика" у него конечно не было, однако после долгих блужданий в отладчике по чрезвычайно сложному и запутанному алгоритму он обнаружил, что подпрограмма, проверяющая легальность, возвращала 0 или 1, в зависимости от исхода проверки. Таким образом, вся защита была отключена после того, как найденную подпрограмму заставили всегда возвращать 0.

Правда, против такого способа снятия защиты есть хорошие приемы обороны: сложное кодирование самой системы защиты (в этом случае взломщик окажется в очень затруднительном положении при попытке корректно подправить исполняемый файл); жуткие ловушки, которые сведут с ума не только ваш трассировщик, но и вас самих; "размазывание" защиты по всей программе, когда вы просто устанете обходить все опасные места и многое другое.

Однако у систем защиты существует второе, намного более опасное и уязвимое место. Его-то я и хочу рассмотреть подробнее.

Как было замечено ранее, компьютерную систему надо воспринимать комплексно. Только такой подход позволит хорошо нападать и хорошо обороняться. Попробуем так и поступить.

Все программы в самом общем случае проделывают одинаковые по своей сути действия — получают информацию из входного потока, обрабатывают ее и уже обработанную информацию направляют в выходной поток. Системы защиты не являются исключением. Вначале они собирают информацию о системе, затем обрабатывают ее (производят идентификацию компьютера, ключевой дискеты, определяют число запусков и т.д.) и, наконец, принимают решение о дальнейших действиях. Таким образом, видно, что блокировать систему защиты можно в трех местах:

1) на этапе сбора информации;
2) на этапе обработки информации;
3) на этапе принятия решения о переходе к дальнейшим действиям.

Как мы убедились раньше, методы взлома, идущие по пути 2 и 3, связаны с тяжелой ручной работой по анализу кода снимаемой защиты. Более интересным, а самое главное, более независимым и универсальным способом является блокировка на этапе сбора информации.

Действительно, нет никакой необходимости в детальном анализе работы снимаемой системы защиты. Достаточно локализовать информационные потоки, на основе которых защита принимает решения, и запомнить их при нормальной работе программы. После этого надо написать загрузчик (достаточно универсальный), который будет запускать несанкционированную копию и создавать для нее иллюзию легальной работы, имитируя входные потоки информации.

Рассмотрим два примера.

Первый пример посвящен снятию защиты с применением ключевой дискеты. Есть хорошие системы, которые формируют ключевые дискеты так, что с них очень тяжело снять копии (для этого надо иметь очень дорогое оборудование, которое невозможно использовать для серийного копирования; в нашей стране таких аппаратов считанные единицы). Ниже приведен метод обхода такой защиты:

1. Пишется специальный драйвер, который полностью протоколирует дисковое прерывание.
2. При загруженном драйвере производится запуск легальной копии.
3. Запоминается полученный протокол обмена с дисководом при опознавании ключевого диска.
4. Формируется новый драйвер, который перехватывает дисковое прерывание, а затем запускает нелегальную копию защищенной программы. Теперь при каждом обращении к ключевому диску драйвер возвращает не реальную, а сохраненную информацию.

Надо отметить, что все указанные действия легко автоматизировать, написав специальную программу, которая и протокол запомнит, и драйвер сформирует. Однако все приведенные выше рассуждения будут верны лишь до тех пор, пока защита получает информацию, действуя через прерывания (самое удивительное, что большинство программ так и поступает). Путь обмана взломщиков, действующих таким путем, прост — надо самим программировать контроллеры внешних устройств через порты и работать с ПЗУ. Но и на это есть управа! Как и раньше, нам на помощь приходит могущественный виртуальный режим.

Метод взлома хитрых защит:

1. Пишется программа, которая переводит процессор в виртуальный режим и запускает легальную копию

взламываемой программы. При этом исследуемой программе запрещаются:
- все операции ввода-вывода (через порты);
- доступ к сегментам памяти, которые могут быть использованы для нужд системы защиты (например, вся область ПЗУ, так как может использоваться его контрольная сумма, область данных BIOS и др.)

2. Если система защиты попытается воспользоваться теми командами и областями памяти, доступ к которым ей запрещен, то через механизм обработки особых ситуаций эти попытки легко обнаруживаются, и снимается протокол обмена информацией.

3. Пишется загрузчик, который запускает нелегальную копию в виртуальном режиме и полностью эмулирует процесс получения информации так, что у системы защиты создается иллюзия штатной работы.

Как и в предыдущем случае, данный метод легко поддается автоматизации. Однако есть смысл написать полуавтоматический взломщик. Это даст возможность опытным путем подобрать минимальный набор ограничений, накладываемых на снимаемую защиту. В противном случае размер файла, содержащего протокол поддержки защиты, может резко возрасти.

Один из методов противодействия наиболее изощренным взломщикам я вижу в использовании виртуального режима в самих системах защиты.

В заключение хочу отметить, что все вышеизложенные рассуждения являются чисто умозрительными (вирусов я не писал, а защитами интересуюсь просто из любопытства), а посему далеко не бесспорными. Однако мне думается, что кое-какие выводы могут оказаться полезными при разработке новых систем защиты как от вирусов, так и от копирования. Я исхожу из предположения, что если один человек придумал что-либо, то и другой может придумать нечто подобное. А подобные проработки вариантов атак помогут заранее разработать предупредительные меры.

Г.Родин

Скандал в Луганске

С 27 по 29 сентября в Луганске проходила выставка-семинар "Компьютер 91", организованная МП "Астра". В выставке участвовали представители следующих фирм:
- Gateway Communications (в лице "Диалог-Сети");
- Microsoft (московская группа);
- Summit Systems;
- ParaGraph;
- Диалог-Экси;
- Диалог;
- Диалог МИФИ

и ряд других.

Организаторы выставки приложили титанические усилия для ее нормального проведения, заручившись предварительной поддержкой местной власти. Однако на практике местные органы власти и правоохранительные органы не обеспечили ни нормального обслуживания участников и гостей, ни их полной безопасности.

Криминогенная обстановка в Луганске такова, что многие из участников смогли ощутить это на себе. Особенно пострадали представители Summit Systems и Gateway Communications. Поразила позиция местного ОМОН, который не только не предупредил возникновение конфликтных ситуаций, но и фактически выступил на стороне преступников, препятствуя вызову правоохранительных органов. При этом был избит представитель фирмы Microsoft. Кроме того была предпринята попытка оклеветать некоторых участников выставки.

На экстренном совместном совещании представителей указанных фирм было принято решение предать максимальной огласке имевший место факт и рекомендовать всем иностранным фирмам воздержаться от контактов с деловыми кругами г. Луганска. Кроме того, заместителю мэра Луганска было заявлено о невозможности выполнения всех контрактов, заключенных на этой выставке, до тех пор, пока власти не смогут обеспечить полной безопасности специалистов.

Луганск, бывший Ворошиловград — основанный в 1795 году промышленный город недалеко от границы Украины и России. Население 500000 человек.

Newsbytes News Network, October 3, 1991

Япония: дебют новой службы, объединяющей устройства вызова и звуковую почту

Orbit System Laboratory в октябре начнет эксплуатацию новой службы, названной "Bell Point", которая содержит в себе систему вызывных устройств и звуковую почту. Пользователи будут звонить на коммутатор Orbit, набирая "7010", а затем номер вызывного устройства. После этого вызываемый абонент сможет позвонить в Orbit и услышать сообщение. Начальная цена 120 йен (90 центов) за три минуты, плюс обычный телефонный тариф NTT. Пользование звуковым почтовым ящиком стоит 210 йен (1,5 доллара) за сообщение. В Токио сейчас 1,5 миллиона человек используют карманные вызывные устройства.

Teleputing/Hotline

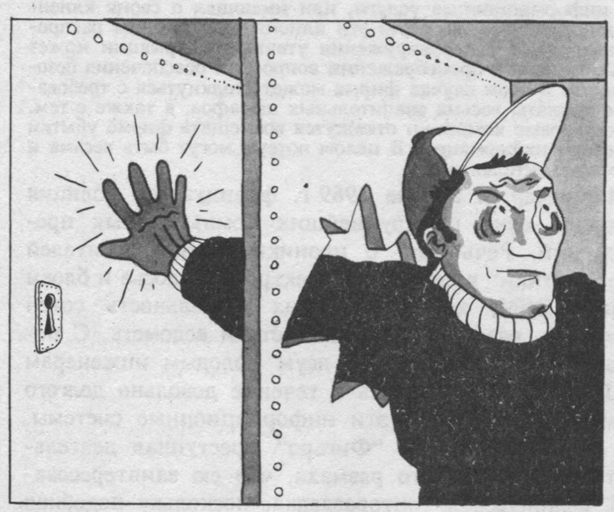
В России распространяется SPRINT

Sprint Networks USSR распространяет свои услуги из Москвы на другие российские города. Открыт центр в Ленинграде, в течение месяца планируется сделать центры в Перми, Самаре, Новосибирске и Хабаровске. К концу 1991 года эта служба будет в десяти русских городах. Все центры будут соединять пользователей с мировыми сетями передачи данных в обход несчастной советской телефонной системы. Sprint заявляет, что информационные агентства Interfax и американское DGL Publishing использовали ее сеть для доставки сообщений во время недавнего кризиса.

Teleputing/Hotline

ELECTRONIC ARTS начала предлагать сопровождение своих программ через оплачиваемую вызывающим линию типа 900 в Северной Америке. Эта фирма производит как игровые, так и серьезные программы. Служба сопровождения предусматривает использование управляемой телефонными тональными кодами базы данных по техническим вопросам. Это будет стоить 95 центов за первую минуту и по 75 центов за каждую следующую минуту — очень дешево для номера серии 900.

Teleputing/Hotline



"По-настоящему безопасной можно считать лишь систему, которая выключена, замурована в бетонный корпус, заперта в помещении со свинцовыми стенами и охраняется вооруженным караулом, — но и в этом случае сомнения не оставляют меня".

Юджин Х. Спаффорд

Основы безопасности компьютерных систем

В связи с прикладываемыми в последнее время государственными органами усилиями в деле компьютеризации страны и "компьютерном ликбезе" населения, а также в связи с изменившимися внутри- и внешнеполитическими условиями, существуют определенные надежды на то, что в недалеком будущем компьютеры для нас перестанут быть зарубежной экзотикой.

Это, конечно, вдохновляет. Но следует понимать, что компьютеризация, кроме очевидных и широко рекламируемых выгод, несет с собой, во-первых, значительные затраты усилий и ресурсов, а, во-вторых, многочисленные проблемы, понятные в настоящее время далеко не всем нашим соотечественникам.

Одной из таких проблем — и притом одной из наиболее сложных проблем — является проблема обеспечения безопасной обработки критичной информации в компьютерных системах.

Здесь и далее под термином "критичная (sensivities) информация" понимается информация с различными грифами секретности; информация для служебного пользования; информация, составляющая коммерческую тайну или тайну фирмы; информация, являющаяся собственностью некоторой организации или частного лица; и так далее.

Вопросы безопасности обработки информации в компьютерных системах пока еще волнуют в нашей стране не слишком широкий круг специалистов. До сих пор эта проблема более или менее серьезно вставала у нас, пожалуй, только перед рядом государственных и военных органов, а также перед научными кругами. Переход нашей страны к рыночной экономике неизбежно повлечет за собой появление многочисленных фирм и банков, эффективная деятельность которых — как показывает зарубежный опыт — практически немыслима без использования компьютеров. Как только ответственные лица этих и других организаций поймут это, перед ними сразу же встанут именно вопросы защиты имеющейся у них критичной информации.

Так что, пока еще есть время, очень стоит серьезно задуматься над имеющимся зарубежным опытом, чтобы не изобретать собственного велосипеда.

Следует сказать, что в настоящее время в нашей стране сложилась достаточно парадоксальная ситуация, когда, с одной стороны, существуют фирмы, организации и частные разработчики, производящие средства защиты компьютерных систем (и, в частности, информации), причем эти фирмы испытывают существенные затруднения с распространением своей продукции (сказывается отсутствие налаженного рынка, рекламы, дилерской сети и т.д.), а, с другой стороны, все значительнее число коммерческих предприятий, банков, бирж, малых предприятий, кооперативов и т.д., использующих компьютерные системы и потому испытывающих настоятельную потребность в защите имеющейся у них информации — или подозревающих о необходимости такой защиты — но не имеющих возможности оценить свои потребности в этой области, не знающих, к кому обратиться за квалифицированной помощью, а, главное, не имеющих возможности целенаправленно приобретать требуемые средства защиты компьютерных систем. Я уже не говорю о том, что достаточно большое количество владельцев компьютерных систем просто-напросто не имеют представления о сути проблемы и использу-

емых в современном мире технологиях ее решения. Конечно, это не их вина, а их беда.

Однозначного объяснения сложившегося парадокса не существует — слишком многие факторы сыграли свою роль в образовании нынешней плачевной ситуации. Однако с большой долей уверенности можно сказать, что одним из таких отрицательных факторов было и остается несовершенство наших законов, регулирующих вопросы человеческой деятельности, связанные с обработкой информации. У нас практически отсутствует какое-либо "информационное" законодательство, если не считать пресловутых ведомственных инструкций и положений об охране государственной и военной тайн. А как быть с тайной коммерческой? А как быть с конфиденциальной информацией о частном лице? А как быть с правами на интеллектуальную собственность? На страже государственных и военных секретов стоят могущественные государственные организации, включая суды. Но кто у нас будет защищать производителя программ от пиратского копирования его продуктов? Кто ответит за разглашение конфиденциальной информации о деятельности негосударственной фирмы? Кто возместит пострадавшим от разглашения информации убытки?

Полагаю, что у многих может возникнуть вопрос: какие убытки? Однако могу сказать, что отсутствие в нашей стране практики оценивания ущерба от утраты (утечки, разглашения и т.п.) информации не означает, что такой практики нигде в мире не существует.

Например, австрийскими специалистами подсчитано, что за 1988 г. экономика Австрии понесла убытки в сумме 1.5 млрд. шиллингов, причиной которых явились сбои и отклонения в работе программ. А по данным Американского Национального Центра Информации по компьютерной преступности за тот же год компьютерная преступность нанесла американским фирмам убытки в размере 500 млн.долларов. В наибольшей степени от нее страдают банки. По словам Ховарда Глассмана, отвечающего за безопасность в Bank of America, компьютерная преступность может стать причиной крушения экономической системы страны.

Можно сказать, что угроза безопасности информации исходит с двух сторон. С одной стороны компьютерным системам — и обрабатываемой в них информации — угрожают "любители" информационных технологий, интересующиеся, прежде всего, собственно компьютерными системами, а не имеющейся в них информацией. К таким лицам относятся так называемые "хакеры" и "крэкеры". Сюда же можно отнести, как ни странно это звучит, непрофессиональных пользователей компьютеров, особенно дилетантов, имеющих некоторые отрывочные сведения о современных компьютерах и уже делающих из этого выводы о собственной "компьютерной образованности".

Любопытным является факт, что в ряде стран, согласно действующему законодательству, к ответственности может быть привлечена, наряду с нарушителем (если его удастся обнаружить и доказать его вину), и "жертва" компьютерного преступления. Например, фирма, предоставляющая некоторые информационные услуги, или имеющая о своих клиентах информацию, которую эти клиенты не склонны распространять, в случае обнаружения утечки информации может быть обвинена в пренебрежении вопросами обеспечения безопасности. В этом случае фирма может столкнуться с требованием выплаты весьма значительных штрафов, а также с тем, что страховые компании откажутся возмещать фирме убытки от потери информации. В целом потери могут быть весьма и весьма значительными.

Например, в апреле 1989 г. французская полиция раскрыла одно из крупнейших компьютерных преступлений. Речь идет о проникновении "любителей информатики" в секретные электронные досье и блоки памяти ЭВМ, контролирующих деятельность сотен компаний, различных министерств и ведомств. С помощью "троянского коня" двум молодым инженерам из Лотарингии удавалось в течение довольно долгого времени проникать в эти информационные системы. По сведениям газеты "Фигаро" преступная деятельность достигла такого размаха, что ею заинтересовалась французская контрразведка, поскольку подобная деятельность может представлять непосредственную угрозу национальной безопасности.

А до этого в октябре 1988 г. в США произошло событие, названное специалистами крупнейшим нарушением безопасности американских компьютерных систем из когда-либо случавшихся. 23-летний студент выпускного курса Корнеллского университета Роберт Таппан Моррис запустил в компьютерной сети ARPANET программу; представлявшую собой редко встречающуюся разновидность компьютерных вирусов — "сетевых червей". (КомпьютерПресс №8'91 и №9'91). В результате атаки был полностью или частично заблокирован ряд общенациональных компьютерных сетей, в частности Internet, CSnet, NSFnet, BITnet, ARPANET и несекретную военную сеть Milnet. В итоге вирус поразил более 6200 компьютерных систем по всей Америке, включая системы многих крупнейших университетов, институтов, правительственных лабораторий, частных фирм, военных баз, клиник, агентства NASA. Общий ущерб от этой атаки оценивается специалистами минимум в 100 миллионов долларов.

Р. Моррис был исключен из университета с правом повторного поступления через год, и приговорен судом к штрафу в 270 тыс.долл. и трем месяцам тюремного заключения.

Но "хакеры" и иже с ними представляют, если можно так выразиться, "внешнюю" угрозу компьютерным системам. Наряду с "внешней" угрозой существует еще и "внутренняя" угроза, исходящая со стороны лиц, работающих в самой фирме. Если говорить более доступно, эта угроза состоит в использовании служебной информации — или возможности ее получения — в личных целях.

Показателен в этом отношении опыт работы Национального Вычислительного Центра Полиции Великобритании (Police National Centre; PNC).

Так, в 1986 г. Управление по рассмотрению исков против полиции (Англия) провело расследование, которое выявило 529 случаев незаконного обращения сотрудников полиции к данным PNC в отношении реги-

страционных номеров владельцев автомобилей, принимавших участие в общенациональной игре "Счастливые номера". Было высказано предположение, что некоторые офицеры полиции предупреждали участников о том, что выбраны номера их автомобилей. В отношении 19 сотрудников полиции были приняты дисциплинарные меры.

Но уже в июне 1987 г. в результате расследования, проведенного Отделом расследования исков Скотланд-Ярда, в соответствии с Законом о защите государственных секретов были привлечены к уголовной ответственности служащий управления полиции г. Хемпстед и частный детектив. Им было предъявлено обвинение в преступном сговоре, имевшем целью незаконное получение данных из PNC и передачу сведений частной сыскной компании.

А в 1988 г. в результате проведенного расследования в Национальном Вычислительном Центре Полиции Великобритании (Police National Centre; PNC) были выявлены запросы с целью определения победителей конкурса номеров автомобилей, организованного автозаправочной компанией. За этим последовало ужесточение правил обращения к PNC, в соответствии с требованиями, изложенными в статьях Закона о защите данных.

В октябре того же года был приговорен к двум годам тюремного заключения по обвинению в краже автомобилей следователь управления полиции Большого Лондона, проработавший в управлении 14 лет. Он использовал свою карточку привилегированного доступа к PNC для получения информации, которая передавалась сообщнику — гражданскому лицу. На основе полученных сведений последний оформлял регистрационные документы на украденные автомобили в Центре регистрации водителей и автомашин, расположенном в г. Суонси.

Следует особо выделить факт наличия в разных странах — и прежде всего в промышленноразвитых и компьютерноразвитых странах — самостоятельной области законодательства, которое многие специалисты называют "компьютерным", хотя на самом деле такого рода законодательство регулирует вопросы, связанные с владением информацией и ее обработкой, и только затем — вопросы, связанные с использованием для обработки этой информации компьютеров. Одним из основополагающих принципов такого законодательства является принцип определения конфиденциальности информации, исходя из реальных свойств самой информации, а не из принятой у нас практики постулирования конфиденциальности, приводящей подчас к совершенно абсурдным ситуациям (когда, например, секретным считается абсолютно чистый лист бумаги только потому, что на нем стоит гриф секретности).

Отметим, что законотворческая практика не стоит на месте. Она активно совершенствуется, в результате чего в ответ на новые виды компьютерных преступлений появляются новые законы, определяющие ответственность за такого рода нарушения.

Так, в 1986 г. Конгрессом США был принят закон о мошенничестве и злоупотреблениях с помощью компьютерной техники. Согласно положениям закона, преступлением считается намеренное несанкционированное проникновение в компьютерную систему, принадлежащую правительству, а также причинение ущерба ЭВМ, в результате которого компьютер не может должным образом выполнять свои функции. Преступлением считается также изменение, уничтожение или предание гласности информации, полученной с помощью несанкционированного доступа к компьютерной системе. Законом предусматривается тюремное заключение сроком до 10 лет.

В январе 1989 г. на основании этого закона к 9 месяцам тюремного заключения и штрафу в 10 тысяч долларов был приговорен 18-летний житель Чикаго (штат Иллинойс) Герберт Зинн. Он был признан виновным в незаконном вторжении в компьютерные системы компании "Америкэн телефон энд телеграф" (AT&T), а также министерства обороны США. Зинн скопировал некоторые программы ЭВМ, среди которых, правда, не было засекреченных.

А в сентябре 1989 г. к 10-летнему тюремному заключению был приговорен Арманд Мур — человек, организовавший "компьютерное ограбление" чикагского "Ферст нэшнл бэнк". Муру и его сообщникам, среди которых были и сотрудники банка, удалось подобрать код к электронной системе банковских операций. Они перевели сумму, превышавшую 69 млн. долларов, из банка в Чикаго в два австрийских банка, расположенных в Вене. После этого мошенники попытались положить деньги уже на свои счета в Америке, но на этой операции были пойманы. Как заявил окружной судья Гарри Лайненуэбер, он не колебался, вынося организатору наиболее суровый приговор за подобное мошенничество.

Тем не менее, наличие соответствующих законов в США не останавливает компьютерных авантюристов. Так, по сведениям газеты "Нью-Йорк Таймс" от 15 января 1989 г. некий Голдис был квалифицирован как "компьютерный бандит", атаковавший компьютерные системы банков, страховых компаний и промышленных фирм с целью обнаружения слабостей в их системах безопасности, которые позволили бы работникам этих учреждений воровать или искажать данные. Голдис является мастером-программистом в среде операционной системы MVS — большой и сложной операционной системы производства фирмы IBM. Его услуги, а также услуги еще ряда специалистов, занимавшихся подобными работами, пользуются все возрастающим спросом. За три года Голдис сумел проникнуть в системы 25 крупных компаний.

Так что же все-таки такое безопасность компьютерных систем?..

1. Безопасность компьютерных систем

Прежде чем перейти к рассмотрению основ обеспечения безопасности при обработке критичной инфор-

мации в компьютерной системе, имеет прямой смысл определить, что мы понимаем под термином "компьютерная система".

Компьютерной системой называется совокупность аппаратных и программных средств, различного рода физических носителей информации, собственно данных, а также персонала, обслуживающего перечисленные выше компоненты. Компьютерная система используется ее владельцем для решения необходимых ему задач.

Теперь можно говорить о безопасности компьютерных систем.

Специалисты признают проблему обеспечения безопасности компьютерных систем одной из наиболее важных и наиболее сложных в области автоматизированной обработки информации.

В настоящее время безопасность компьютерных систем стала самостоятельным направлением научных исследований и разработок со своей терминологией, методами, моделями и т.д.

Рассмотреть проблему безопасности целиком в масштабах относительно небольшой статьи не представляется возможным. По данной тематике написаны весьма объемистые монографии, но нет и намека на то, что неосвещенных вопросов и нерешенных проблем в этой области стало меньше. Во многом это определяется темпами научно-технического прогресса, поставляющего человечеству все более новые компьютерные технологии. Это не только ставит новые проблемы обеспечения безопасности, но и представляет, казалось бы, решенные вопросы и проблемы зачастую в новом, порой неожиданном, ракурсе.

Моя цель поэтому значительно скромнее: ознакомить пользователей компьютерных систем с некоторыми основами проблемы обеспечения безопасности, с рядом общепринятых терминов и — самое главное — попытаться продемонстрировать серьезность и актуальность этой проблемы для всех, кто так или иначе связан с использованием компьютеров.

Итак, несколько основных определений.

ЭКСПОЗИЦИЕЙ называется форма возможной потери или ущерба для компьютерной системы. Например, экспозициями считаются неавторизованный доступ к данным или противодействие авторизованному использованию компьютерной системы.

УЯЗВИМОСТЬЮ называется некоторая слабость системы безопасности, которая может послужить причиной нанесения компьютерной системе ущерба.

АТАКОЙ называется действие некоторого субъекта компьютерной системы (пользователя, программы, процесса и т.д.), использующего уязвимость компьютерной системы для достижения целей, выходящих за пределы авторизации данного субъекта в компьютерной системе. Более просто: если, например, пользователь не имеет права на чтение некоторых данных, хранимых в компьютерной системе, а ему очень хочется, и поэтому он предпринимает ряд известных ему нестандартных манипуляций, обеспечивающих доступ к этим данным (в случае отсутствия или недостаточно надежной работы средств безопасности), либо завершившихся неудачей (в случае надежной работы средств безопасности) — тем самым этот пользователь (иногда его называют "захватчиком") предпринимает в отношении компьютерной системы атаку.

УГРОЗОЙ для компьютерной системы являются условия, представляющие потенциальную возможность нанесения ущерба компьютерной системе. Атаки — частный вид угроз, также как и стихийные бедствия, человеческие ошибки, программные сбои и т.д.

И, наконец, УПРАВЛЕНИЕМ в терминологии безопасности называется защитный механизм (действие, устройство, процедура, технология и т.д.) уменьшающий уязвимость компьютерной системы.

Следует понимать, что ущерб компьютерной системы — понятие также достаточно широкое. Ущербом считается не только явное повреждение какого-либо из компонентов компьютерной системы (например, кувалдой — да по терминалам!), но и приведение компонентов системы в неработоспособное состояние (например, обесточивание помещения, в котором находятся аппаратные средства), и различного рода утечки информации (например, незаконное копирование программ, получение конфиденциальных данных), и изменение некоторых физических и логических характеристик компьютерной системы (например, неавторизованное добавление записей в системные файлы, повышение загрузки системы за счет запуска дополнительного неучтенного процесса/программы и т.д.).

Определение возможного ущерба компьютерной системы — дело весьма сложное, зависящее от многих условий. Например, от того, признается ли юридически в данной стране так называемая интеллектуальная собственность или общеизвестный Copyright, рассматриваются ли судами иски по возмещению морального ущерба, понесенного некоторым лицом или организацией в результате разглашения третьей стороной конфиденциальной информации и т.д. Уже эти вопросы достаточно наглядно демонстрируют, насколько неподготовленным вступает в современный компьютерный мир — столь желанный и столь небезопасный — наше Отечество. Ведь здесь владельцы компьютеров в большинстве своем считают диким приобретение программы за деньги у разработчика или его представителя, если есть возможность раздобыть ту же программу полулегально, но бесплатно — просто скопировав ее у того, кто ее имеет, иногда даже без разрешения. И ведь так — практически во всех вопросах, касающихся использования компьютеров! Многие даже понятия не имеют, что подобные анархистские действия не просто и не только безнравственны, но и опасны!

Предвижу вопросы: к чему такая патетика? Какой ущерб? Какая опасность? Опасность весьма очевидна — можно запросто испортить, а то и вообще лишиться своей компьютерной системы или какого-либо из ее компонентов (чаще всего — данных).

А насчет ущерба... Прямого ущерба от того, что некто прочитал у вас какие-то данные или сделал что-то

в вашей системе, действительно, может и не быть. Но, допустим, кто-то из пользователей вашей системы сумел запустить некоторый процесс в обход принятых в вашей системе правил (например, не зарегистрировав процесс у оператора). Стоимость затраченного на выполнение этого процесса вашего машинного времени, которое не будет оплачено (процесс-то не зарегистрирован) — это как, ущерб для вас или нет?

Еще маленький пример: некто в силу сложившихся обстоятельств смог получить доступ к вашей автоматизированной системе учета кадров, "полазил" по ней и узнал, например, список лиц, находящихся на учете у нарколога. Возможные неприятности этих людей, которые могут произойти вследствие разглашения этих конфиденциальных сведений — это ущерб для вашей организации?

Таких примеров можно привести очень много. Я ведь даже не упомянул еще о деятельности организаций, занятых вопросами, составляющими государственную или военную тайну; я не затронул проблему возможных нарушений работы компьютеров, управляющих процессами химического производства или, допустим, ядерным реактором; я не сказал об обеспечении сохранности коммерческой тайны, либо обеспечении приоритета в научно-исследовательских разработках.

Можно с уверенностью сказать, что везде, где используются компьютеры (а дело идет к тому, что они будут использоваться везде), существует потенциальная угроза нанесения ущерба — прямого или косвенного — законным владельцам и законным пользователям этих компьютеров.

С другой стороны, заслуживает внимания вопрос о стоимости самой информации. Это далеко не простой вопрос. В самом деле, достаточно сложно представить, что сотрясение воздуха или электромагнитное колебание может чего-либо стоить. Однако, в мировой компьютерной практике принято, что информация стоит ровно столько, сколько стоит ущерб от ее потери в сочетании с затратами на ее восстановление.

Если говорить более точно, то это правило применяется вообще в отношении любого компонента компьютерных систем, а не только хранимой или обрабатываемой в них информации.

Вопросы, касающиеся безопасности компьютерных систем, можно условно разделить на следующие группы:

1. Вопросы обеспечения физической безопасности компонентов компьютерной системы. Сюда относятся вопросы защиты компьютерных систем от пожара, затопления, других стихийных бедствий, сбоев питания, кражи, повреждения, задымления и т.д.
2. Вопросы обеспечения логической безопасности компонентов компьютерных систем. Сюда относятся вопросы защиты компьютерных систем от неавторизованного доступа, от умышленных и неумышленных ошибок в действиях людей и программ, которые могут привести к ущербу и т.д.
3. Вопросы обеспечения социальной безопасности компонентов компьютерных систем. Сюда относятся вопросы разработки законодательства, регулирующего применение компьютеров и определяющего порядок расследования и наказания нарушений безопасности компьютерных систем; принципы и правила такой организации обслуживания пользователей в компьютерных системах, которая уменьшала бы риск нарушения безопасности компьютерных систем и т.д.
4. Вопросы обеспечения этической безопасности компонентов компьютерных систем.

Возможно, кому-то это покажется не столь важным, но многие специалисты считают, что в обеспечении безопасности компьютерных систем немалую роль играют вопросы выработки у пользователей определенной дисциплины, а также формирование определенных этических норм, обязательных для выполнения всеми, кто работает с компьютерами. Например не так давно эксперты Национального Научного Фонда США предприняли попытку выработать своеобразный "кодекс поведения" компьютерного специалиста. В частности, указывалось, что неэтичными следует считать любые умышленные или неумышленные действия, которые:

1) нарушают нормальную работу компьютерных систем;
2) вызывают дополнительные затраты ресурсов (машинного времени, полосы передачи и т.д.);
3) нарушают целостность хранимой и обрабатываемой в компьютерных системах информации;
4) нарушают интересы законных пользователей;
5) вызывают незапланированные затраты ресурсов на ведение дополнительного контроля, восстановление работоспособности систем, уничтожение последствий нарушения безопасности систем и т.д.

Как следует из определения компьютерной системы, основными ее компонентами являются аппаратные средства, математическое (в том числе, программное) обеспечение и данные (информация).

С точки зрения теории существует "всего лишь" четыре типа угроз этим компонентам:

- ПРЕРЫВАНИЕ: при прерывании компонент системы утрачивается (например, в результате похищения), становится недоступным (например, в результате блокировки — физической или логической), либо теряет работоспособность;
- ПЕРЕХВАТ: некоторая третья неавторизованная сторона получает доступ к компоненту. Примерами перехвата являются незаконное копирование программ и данных, неавторизованное чтение данных из линий связи компьютерной сети и т.д.;
- МОДИФИКАЦИЯ: некоторая третья неавторизованная сторона не только получает доступ к компоненту, но и манипулирует с ним. Например, модификациями являются неавторизованное изменение данных в базах данных или вообще в файлах компьютерной системы, изменение алгоритмов используемых программ с целью выполнения некоторой дополнительной незаконной обработки. Иногда модификации обнаруживаются достаточно быстро (если

фикации обнаруживаются достаточно быстро (если не сразу), но более тонкие модификации могут оставаться необнаруженными весьма длительное время;

- ПОДДЕЛКА: захватчик может добавить некоторый фальшивый процесс в систему для выполнения нужных ему, но не учитываемых системой, действий, либо подложные записи в файлы системы или других пользователей.

Например, зная формат записи в файле, на основании которого в вашей организации начисляется зарплата, вполне можно занести в этот файл поддельную запись. При начислении зарплаты эта запись будет обрабатываться (если все сделано достаточно тонко) наравне с законными записями — но кто будет получать деньги, причитающиеся несуществующему работнику? Догадываетесь?

перехват
прерывание (потеря) — данные — модификация, подделка
прерывание (отказ от пользы), перехват (кража) — аппаратные средства
математическое/программное обеспечение — модификация, прерывание (удаление), перехват

На первый взгляд столь малый список может неоправданно ободрить пользователей: может и не так страшен черт, как его малюют. Чего там — всего-то четыре вида опасности. Но если бы вы только знали, в сколь разных и сколь многочисленных образах могут проявляться эти четыре угрозы!

Так что легкой победы не будет. Если вы хотите быть уверенными в том, что ваши данные никто не прочитает, не сотрет, не исказит, не украдет ваш компьютер, не воспрепятствует вашей работе и еще много-много всякого рода "не..." — готовьтесь к планомерной, трудной и нудной обороне.

Опять же, с точки зрения теории, безопасность любого компонента компьютерной системы складывается из обеспечения трех его характеристик: секретности, целостности и доступности.

Секретность состоит в том, что компонент системы доступен только авторизованным субъектам системы (пользователям, программам и т.д.) Для остальных — неавторизованных — субъектов этот компонент как бы не существует (лучше сказать, он им "невидим").

Целостность компонента предполагает, что компонент может быть модифицирован только авторизованным для этого субъектом (подразумевается, что этот субъект осуществляет все модификации корректно и не ставит целью осложнение собственной жизни путем искажения собственных данных или поломки собственного устройства). Иными словами целостность — гарантия правильности (работоспособности) компонента в любой момент времени. Отметим, что под модификацией подразумеваются операции записи, обновления, изменения состояния, удаления и создания.

Доступность предполагает действительную доступность компонента авторизованному субъекту, т.е. авторизованный субъект может в любой момент без особых проблем получить доступ к необходимому компоненту.

Таковы основные теоретические посылки, необходимые для понимания дальнейшего изложения в частности, и всей проблемы обеспечения безопасности компьютерных систем в целом.

Дальнейшее изложение будет посвящено разъяснению необходимости учета вопросов обеспечения безопасности в повседневной работе. В основном будут продемонстрированы не столь очевидные для пользователей логические угрозы безопасности, а также пояснена роль некоторых основных механизмов обеспечения безопасности.

И.Моисеенков

По материалам:
D.Tassel, "Computer Security Management", Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1972.
D.Denning, "Cryptography and Data Security", Purdue University, AddisonWesley Publishing Company, 1982.
D.Davies, W.Price, "Security Computer Networks", John Wiley & Sons, 1984.
C.Pfleeger, "Security in Computing", 1988, University of Tennessee and Trusted Information Systems, Inc..
E.Spafford, "The Internet Worm Programm: An Analysis", ACM Commitee Report, 1989.
Datapro Report, January 1989.
P.Fites, P.Johnston, M.Kratz, "The Computer Virus Crisis", 1989.
"Computer & Security", 8(1989).
"КомпьютерПресс", N1-2 (1989), N3-10 (1990).

(Продолжение следует)

4 Зак. 2246

Одной из первых проблем, возникающих после приобретения IBM-совместимого персонального компьютера, является его русификация, то есть обеспечение возможности ввода русского текста с клавиатуры и просмотра его на экране в различных режимах. В данной статье обсуждаются вопросы, связанные с русификацией IBM PC. Надеемся, что ее с интересом прочтут и квалифицированный программист, и новичок, каждый из которых найдет здесь полезную для себя информацию. •

Русский драйвер экрана и клавиатуры

Вначале совершим небольшой исторический экскурс и освежим хронологию развития видеоадаптеров IBM-совместимых персональных компьютеров.

Первым видеоадаптером семейства IBM-совместимых персональных компьютеров был MDA (Monochrome Display Adapter — адаптер черно-белого дисплея). Он имел всего 4 Кбайта видеопамяти и, следовательно, не поддерживал графические режимы работы. Поэтому русификация его была возможна исключительно на аппаратном уровне путем перепрограммирования ППЗУ знакогенератора. Несколькими годами позже появились видеоадаптеры CGA (Color Graphics Adapter — цветной графический адаптер) и Hercules (графический адаптер фирмы Hercules). Адаптер CGA имел 32 Кбайта видеопамяти и мог работать как в текстовом, так и в графическом режимах, однако возможности программной загрузки знакогенератора в нем еще отсутствовали. Адаптер Hercules явился потомком MDA. Этот монохромный адаптер с 32 Кбайтами видеопамяти поддерживал разрешение 720x348 точек черно-белого изображения. Высокое разрешение Hercules обеспечило возможность получения качественного экранного шрифта. Однако его BIOS, к сожалению, не поддерживал программной загрузки знакогенератора, поэтому русификация Hercules также была возможна только на уровне перепрограммирования ППЗУ знакогенератора.

Тем не менее, с видеоадаптерами CGA и Hercules можно работать на русском языке в графических режимах без аппаратного перепрограммирования ПЗУ, что и реализовал для CGA Е.Веселов в своем популярном редакторе "Лексикон". Другим примером аналогичного текстового процессора, обладающего гораздо большими возможностями, является ChiWriter фирмы Horstmann Design Software. "Лексикон" и ChiWriter на CGA работают с разрешением 640x200 точек черно-белого изображения. Хотя это и лучшее разрешение, которое способен обеспечить CGA, все же длительная работа в таком режиме приводит к сильному утомлению глаз. Этот эффект хорошо известен пользователям, которые все еще работают с CGA. Таких, к сожалению, немало, ибо до сих пор большинство ПЭВМ, выпускаемых в нашей стране (в это число не входят IBM-совместимые персональные компьютеры, собираемые в СССР из зарубежных комплектующих), оснащаются CGA. Впрочем, ЕС-1840, например, имеет в качестве адаптера некий гибрид. В общем-то это самый обыкновенный CGA, но с возможностями загруз-

ки знакогенератора, хотя и нестандартным способом. Тем не менее, работа с CGA весьма утомительна для глаз, а возможности этого видеоадаптера слишком ограничены, поэтому через некоторое время был разработан адаптер EGA (Enhanced Graphics Adapter — усовершенствованный графический адаптер).

Стандартный вариант видеоадаптера EGA имеет 256 Кбайт видеопамяти и набор функций для поддержки операций со знакогенератором. Возможности EGA таковы: 8 видеостраниц в текстовых режимах работы, и 2 видеостраницы в графическом режиме высокого разрешения (640x350 точек, 16 цветов). Как и в Hercules и MDA, передача изображения дисплею здесь осуществляется цифровым сигналом. Существует много разновидностей этого адаптера, выпускаемых различными фирмами (HEGA, SEGA, VEGA, PEGA...). Как правило, они отличаются от стандартного EGA возможностями работы в нестандартных режимах. Однако монитор при этом должен соответствовать адаптеру, иначе вы не сможете работать в дополнительных режимах адаптера. Именно EGA является на текущий момент наиболее распространенным адаптером IBM-совместимых персональных компьютеров, хотя в последнее время за рубежом его вытесняет VGA (Video Graphics Array — видеографический массив).

Видеоадаптер VGA отличается от EGA гораздо большими возможностями и принципиально иным устройством на уровне регистров адаптера. Здесь осуществлен возврат к аналоговой форме передачи видеосигнала, что дало возможность одновременного отображения на экране 256 цветов из 256 К возможных. Наилучшее разрешение VGA при 16 цветах — 640x480 точек. VGA — это первый видеоадаптер IBM-совместимых персональных компьютеров, имеющий квадратную точку. Это значит, что окружность, построенная по математической формуле, на VGA будет действительно окружностью, а не эллипсом, как на всех других адаптерах. Матрица символов VGA имеет размер чуть больший, чем на EGA — 8x16, что связано с повышенным, по сравнению с EGA, разрешением по вертикали. BIOS VGA значительно расширен, но, с точки зрения драйвера экрана, VGA отличается от EGA только размером матрицы и, следовательно, числом байт, необходимых для кодирования одного символа.

С клавиатурой все обстоит проще. Существует два основных стандарта клавиатуры: стандартная клавиатура (84 клавиши) и расширенная клавиатура (101/102 клавиши). В последнее время появились также различные вариации на тему расширенной клавиатуры, содержащие в некоторых случаях до 132 клавиш. Однако все дополнительные клавиши лишь дублируют основные и собственных уникальных кодов не вырабатывают. Компьютеры IBM PC-XT ранее комплектовались, в основном, стандартной клавиатурой, а IBM PC-AT — расширенной. Сегодня на компьютерном рынке стандартная 84-клавишная клавиатура стала экзотикой, но у пользователей она еще встречается довольно часто.

На первый взгляд кажется, что два варианта клавиатуры отличаются незначительно — всего лишь числом и расположением клавиш, но на самом деле это не так. Их главное различие — в присутствии в расширенной клавиатуре специального контроллера, чего лишена стандартная клавиатура. Существенную роль играет и разное количество клавиш на этих клавиатурах, а также то, что аппаратно клавиатуры устроены принципиально по-разному, хотя расширенная клавиатура и обеспечивает все возможности стандартной.

Собственно, проблема работы в DOS на родном языке пользователя (в частности, на русском) возникла тогда, когда IBM-совместимые персональные компьютеры стали, фактически, стандартом персональных компьютеров во всем мире.

Человек, не посвященный во внутреннее устройство IBM PC, может спросить: "А откуда появилась возможность одновременной работы на двух языках на одном мониторе?". Такая возможность появилась после принятия 8-битной кодировки символов ASCII. На старых моделях мини- и микроЭВМ применялась 7-битная кодировка, что ограничивало пользователя в выборе рабочего языка. Одновременно в работе можно было использовать только один язык. Возможность использования другого шрифта определялась программированием ПЗУ знакогенератора монитора. Причина этого заключается в следующем: при помощи 7 бит можно закодировать только 128 символов. При этом 2*26 = 52 символа занимает английский алфавит, 10 символов приходится на цифры, остальные — на знаки препинания и специальные символы. Очевидно, что места для алфавита другого языка не остается.

Положение изменилось коренным образом, когда была принята 8-битная кодировка ASCII, которая позволяет использовать 256 символов. При этом первые 128 символов совпадают со старой семибитной кодировкой (для совместимости снизу вверх), а остальными 128 комбинациями кодируются дополнительные символы. Обычно верхние 128 кодов (так называемая внешняя часть знакогенератора) содержат символы псевдографики, часть греческого алфавита и прочие символы, специфичные для других алфавитов.

Для большинства стран, использующих латинский, либо модифицированный латинский алфавит, во внешней части знакогенератора имеются все необходимые символы. Поэтому нет необходимости в загрузке знакогенератора вообще. Для настройки на конкретный язык в DOS предусмотрены несколько утилит (GRAPHTABL, KEYB, NLSFUNC). Естественно, ни одна из них не позволяет работать с кириллицей, поскольку до недавнего времени не существовало русифицированной версии Microsoft DOS. Для получения возможности работы на русском языке необходимо сделать два шага: во-первых, запрограммировать определенным образом знакогенератор адаптера (или перепрограммировать ППЗУ, если адаптер не способен загружать знакогенератор), а во-вторых, обеспечить возможность переключения клавиатуры на другую рас-

кладку для ввода символов второй половины таблицы знакогенератора.

Следует отметить, что существует несколько вариантов кодировки второй половины таблицы знакогенератора русскими буквами. На данный момент стандартом является так называемая "модифицированная альтернативная кодировка ASCII", на которую и рассчитано большинство программ, использующих вторую половину знакогенератора. Но, к сожалению, в нашей стране все еще выпускаются ПЭВМ с основной и болгарской кодировками ASCII (Нейрон, Искра и другие), а также с раскладками клавиатуры, отличающимися от стандартной QWERTY/ЙЦУКЕН — QWERTY/ЯВЕРТИ, JCUKEN/ЙЦУКЕН. Таким образом, для обеспечения нормальной работы на русском языке пользователь должен точно знать следующее:

- какой видеоадаптер подключен к компьютеру;
- какая клавиатура подключена к компьютеру;
- какая кодировка ASCII применяется на компьютере;
- какая раскладка алфавита на клавиатуре.

На основании этой информации можно либо сконфигурировать имеющуюся у пользователя систему, например, BETA, либо модифицировать текст драйвера экрана и клавиатуры. Следует отметить, что в первых программах поддержки кириллицы на IBM-совместимых персональных компьютерах использовался не объединенный драйвер экрана и клавиатуры, а два драйвера — один для работы с адаптером, другой для управления клавиатурой. Для обеспечения работы с любой конфигурацией экрана и клавиатуры система BETA, например, требует набора из 6 драйверов, которые описывают шрифты, их кодировку, раскладку клавиатуры и другие параметры. Но все же большинство применяемых драйверов — комбинированные, хотя для специальных целей иногда применяются раздельные программы для экрана и клавиатуры.

Таким образом, стандартной конфигурацией аппаратного комплекса, обсуждаемой в дальнейшем, является следующая: видеоадаптер EGA/VGA, альтернативная модифицированная кодировка ASCII, раскладка клавиатуры QWERTY/ЙЦУКЕН.

Как уже было отмечено выше, драйвер поддержки кириллицы состоит из двух частей — программы поддержки экранных шрифтов и программы управления клавиатурой.

Начнем с драйвера экранного шрифта. Вообще говоря, если вы не намерены изменять текущий видеорежим (пользоваться функцией 00h прерывания 10h), то драйвер экрана как таковой не нужен. Достаточно просто загрузить русские символы в знакогенератор, используя функцию 11h прерывания 10h, после чего вы сможете работать с кириллицей. Однако, любая программа, переводящая адаптер в другой видеорежим, вызовет функцию 00h прерывания 10h, чем уничтожит загруженный фонт. Более того, пользуясь таким подходом, вы никогда не получите русского шрифта в графических режимах работы (если, разумеется, вызываемая программа сама не обеспечивает поддержку кириллицы). Поэтому для непрерывной поддержки русских экранных фонтов в любых режимах работы видеоадаптера необходим специальный драйвер.

Как правило, в знакогенератор загружаются два фонта: 8x14 для текстовых режимов работы EGA (или 8x16 для VGA) и 8x8 для работы в графических режимах. Среди прочих функций знакогенератора присутствует функция, сообщающая информацию о фонтах, используемых для отображения символов. Для того, чтобы запрашивающая программа получала правильную информацию, драйвер должен контролировать и эту функцию и сообщать верные адреса фонтов. Таким образом, в состав объединенного драйвера, который поддерживает работу с кириллицей и латинским алфавитом, должны входить следующие обработчики прерываний:

- обработчик прерывания 09h. Предназначен для проверки нажатия ключевой клавиши и переключения драйвера с одного алфавита на другой (включая раскладку клавиатуры);
- обработчик прерывания 16h. Предназначен для проверки граничных условий необходимости перекодировки символов латинского алфавита в кириллицу и выполнения самой перекодировки;
- обработчик прерывания 10h. Предназначен для отслеживания всех функций BIOS EGA, связанных со знакогенератором, и обеспечения правильного вывода информации на экран и сообщения информации о загруженных фонтах;
- таблица, содержащая конфигурацию (рисунки) всех символов фонта 8x14;
- таблица, содержащая конфигурацию всех символов фонта 8x8.

Предлагаемый вашему вниманию драйвер AD_DRV реализует сформулированные выше правила и рекомендации. Он предназначен для работы на EGA и поддерживает альтернативную модифицированную кодировку ASCII с раскладкой клавиатуры QWERTY/ЙЦУКЕН. В данной конфигурации драйвер рассчитан на расширенную клавиатуру; переключение с одного алфавита на другой производится нажатием и отпусканием правой клавиши Control.

Существует также много других вариантов переключения драйвера с одного алфавита на другой, например, нажатием обеих клавиш Shift, одной из клавиш Alt, комбинаций Alt+Shift, Alt+Enter и др. Однако следует признать, что многоклавишные комбинации неудобны, так как при частых переключениях чрезвычайно утомительно набирать такой "аккорд". Вероятно, вы сами работали с такими драйверами, и убедились, что очень трудно нажимать сложную комбинацию несколько раз в минуту. Правая клавиша Control находится в правом нижнем углу клавиатуры, что весьма удобно, так как, во-первых, ее легко достать мизинцем правой руки, а, во-вторых, ее расположение совпадает с расположением клавиши CapsLock на стандартной клавиатуре. Таким образом, если вы пользуетесь адаптером EGA и расширенной

клавиатурой, то драйвер не требует модификации, а если вы пользуетесь стандартной клавиатурой, то потребуется заменить обработчик прерывания 09h. Ниже приводится полный текст обработчика прерывания 09h для стандартной клавиатуры (84 клавиши). Здесь переключение алфавита будет осуществляться нажатием клавиши CapsLock, которая расположена примерно в том же месте, что и правый Control на расширенной клавиатуре.

```
int09 label byte
      cli
      push AX          ; Сохранение
      push BX          ;  регистров
      push DS
      xor  BX,BX       ; Очистить BX
      mov  DS,BX       ; Получить байт статуса
      mov  BX,0418h    ;  клавиатуры из
      mov  AH,[BX]     ;  BIOS Data Area
      pushf            ; Сохранить флаги
      db   09ah
old9o dw   0           ; В этих переменных находится
old9s dw   0           ;  старый адрес прерывания 09h
      mov  AL,[BX]
      and  AX,4040h    ; Проверка байта статуса AX,0040h
                       ;  клавиатуры на CapsLock
      jnz  fut1
      mov  CS:Pressed,1
      jmp  bye         ; CapsLock не нажат — закончить
fut1:
      cmp  AX,4000h    ; Вторая проверка на CapsLock
      jnz  fut2        ; CapsLock не отпущен — закончить

      mov  AX,DS:[417h]
      and  AX,0BFh       ; Отключить
      mov  DS:[417h],AX  ;  состояние CapsLock
      cmp  CS:Pressed,1
      jnz  fut2
      not  CS:Cyrillic ; Переключить драйвер в
                       ;  альтернативное состояние
fut2:
      mov  CS:Pressed,0
bye:
      pop  DS          ; Восстановить регистры
      pop  BX
      pop  AX
      iret
```

Если вы используете клавиатуру с раскладкой, отличающейся от QWERTY/ЙЦУКЕН, то вам понадобится изменить кодировку клавиш. Кодировка содержится в двух таблицах: Ordtbl — для работы в режиме вывода строчных букв, и Cpstbl — для работы в режиме вывода прописных букв (включен индикатор CapsLock). Устройство таблиц понятно из их внешнего вида.

В данной статье не приводится таблиц фонтов, так как, во-первых, это значительно увеличило бы объем статьи, а во-вторых, нет смысла вручную составлять таблицу фонтов, поскольку необходимый код можно сформировать при помощи любого редактора фонтов. Наиболее известным и проверенным редактором фонтов для видеоадаптеров EGA/VGA является система EVAfont (автор П.Квитек). Эта программа является freeware и доступна на BBS СП Диалог (телефон (095)329-21-92 для звонков через модем, или (095)328-12-27 для обычных звонков). В комплект EVAfont входит набор фонтов 8x8 (CGA, графические режимы низкого разрешения EGA/VGA), 8x14 (EGA) и 8x16 (VGA). Вы можете выбрать подходящие фонты, а затем при помощи функции "Output MASM" создать необходимые файлы конфигурации фонтов 8x8 и 8x14/8x16. Если же вас не удовлетворяет ни один из имеющихся в наборе фонтов, то, пользуясь редактором фонтов, вы можете создать свой собственный.

Ниже следует краткое описание работы подпрограмм-обработчиков прерываний драйвера AD_DRV.

Обработчик прерывания 09h

Прерывание 09h генерируется каждый раз, когда нажимается или отпускается любая клавиша клавиатуры. Слежение за прерыванием 09h дает возможность более точно проверить условия переключения драйвера. Рассмотрим, как это делается. Прерывание 09h не возвращает никакого кода, оно просто сигнализирует о том, что нажата или отпущена клавиша. Для получения кода нажатой клавиши необходимо считать данные из порта клавиатуры 60h. После этого программа обрабатывает принятый scan-код. Для начала необходимо проверить префикс расширенной клавиатуры, а затем — scan-код правой клавиши Control. Если правый Control нажат, то в переменную Pressed засылается единица. При следующем вхождении осуществляется проверка, является ли текущий scan-код кодом отпускания правой клавиши Control, а также проверка значения Pressed. Таким образом, можно определить, нажимались ли какие-либо промежуточные клавиши во время удержания правой клавиши Control в нажатом состоянии. Предложенный алгоритм позволяет использовать правый Control и как клавишу-модификатор, и как переключатель драйвера из одного состояния в другое. Если правый Control был применен как переключатель, то обработчик инвертирует переменную Cyrillic, которая служит индикатором текущего режима работы.

Обработчик прерывания 16h

Прерывание 16h позволяет принять ASCII код (или расширенный ASCII код) нажатой клавиши, а также выполняет другие операции с клавиатурой. За прием кода отвечают функции 00h, 01h, 10h и 11h. Все эти функции необходимо отследить, однако, вследствие их различия, приходится вводить две подпрограммы обработки — func0 для функций 00h/10h и func1 для функций 01h/11h. В обеих подпрограммах проверяется значение переменной Cyrillic. Если переменная отлична от нуля, то вызывается процедура translate, которая и выполняет собственно перекодировку с учетом того, что знаки препинания, цифры, символы забоя (BackSpace) и пробела в перекодировке не нуждаются.

Обработчик прерывания 10h

Прерывание 10h обеспечивает работу с видеосистемой. Обработчик необходим для контроля за функцией 00h (переключение видеорежима), а также за функциями BIOS EGA/VGA, связанными со знакогенератором адаптера. Если произошло переключение видеорежима, следовательно, предыдущий фонт утерян, и его необходимо загрузить вновь. При этом следует учитывать текущий видеорежим и загружать фонт соответствующего размера — 8x8 или 8x14/8x16. Необходимо также отличать текстовые режимы от графических, так как за загрузку фонта в разных режимах отвечают различные функции BIOS. Обработчик прерывания 10h осуществляет необходимые проверки, после чего производится загрузка соответствующего фонта и возврат из прерывания.

Следует отметить, что драйвер AD_DRV перехватывает еще два вектора прерываний — 44h и 1Fh. Эти векторы указывают на фонты для текстового и графического режимов. Драйвер осуществляет переустановку векторов 44h и 1Fh таблицы прерываний на загружаемые фонты.

Программа установки драйвера в памяти стандартна: сначала проверяется наличие драйвера в памяти, после чего производится освобождение DOS Environment, а также сохранение и установка векторов прерываний. В случае, если драйвер уже присутствует в памяти, или к компьютеру подключен видеоадаптер, отличный от EGA или VGA, выдается соответствующее предупреждение и осуществляется выход из программы. При указании в командной строке соответствующих ключей (см. текст программы) загруженный ранее драйвер выгружается из памяти.

```
; Драйвер AD_DRV поддержки кириллицы на экране и
; клавиатуре IBM-PC.
; Данная версия предназначена для работы с видеоадаптером
; EGA и подразумевает использование альтернативной\
; кодировки ASCII и раскладки клавиатуры
; QWERTY/ЙЦУКЕН.
; При разработке драйвера AD_DRV использовался
; ассемблерный код программ EN_DRV Свиридова И.А. и
; EGAGA Козлова А.В.
;
; Program was written by JVK Software.  All rigths reserved.
; Июнь 1991 г.
;
; Эта программа является freeware и может передаваться в
; неизмененном виде всем пользователям в некоммерческих
; целях.
; Если у Вас есть замечания и предложения по работе
; программы AD_DRV, свяжитесь с автором по телефонам:
; 135-99-07 (Москва), 477-13-24 (Киев)
; Кравацкий Юрий Всеволодович

CODE segment word
  assume CS:CODE
  org    100h

Border      equ  1            ; Цвет рамки экрана

pushrs  macro               ; Макроопределение
  push  AX                  ; для сохранения
  push  BX                  ; изменяемых регистров
  push  CX
  push  DX
  push  ES
  push  BP
endm

poprs  macro                ; Макроопределение
  pop   BP                  ; для восстановления
  pop   ES                  ; изменяемых регистров
  pop   DX
  pop   CX
  pop   BX
  pop   AX
endm

start:
  jmp   install             ; Перейти к блоку установки

; Функции и данные, связанные с прерываниями 09h и 16h
;   (клавиатура)

Ext        db  0            ; Признак расширенной клавиатуры
Pressed    db  0            ; Признак нажатия правой
                            ;   клавиши Control
Cyrillic   db  0            ; Признак включенной кириллицы

; Таблица символов для работы в режиме вывода
; строчных букв
Ordtbl     label            byte
  db   32 ,'1','Э','3','4','5','7','э','9','0'
  db   '8',' + ','б','-','ю','/',')','!','"',' #'
  db   '$',':',',',',','.',';','(','Ж','ж','Б',' = '
  db   'Ю','?','2','Ф','И','С','В','У','А','П'
  db   'Р','Ш','О','Л','Д','Ь','Т','Щ','З','Й'
  db   'К','Ы','Е','Г','М','Ц','Ч','Н','Я','х'
  db   '\','ъ','б','_','"','ф','и','с','в','у'
  db   'а','п','р','ш','о','л','д','ь','т','щ'
  db   'з','й','к','ы','е','г','м','ц','ч','н'
  db   'я','Х','|','ъ','~',127

; Таблица символов для работы в режиме CapsLock
; (вывод прописных букв)
Cpstbl     label            byte
  db   32 ,'1','э','3','4','5','7','э','9','0'
  db   '8',' + ','Б','-','Ю','/',')','!','"',' #'
  db   '$',':',',',',','.',';','(','ж','Ж','б',' = '
  db   'ю','?','2','Ф','И','С','В','У','А','П'
  db   'Р','Ш','О','Л','Д','Ь','Т','Щ','З','Й'
  db   'К','Ы','Е','Г','М','Ц','Ч','Н','Я','Х'
  db   '\','ъ','б','_','"','ф','и','с','в','у'
  db   'а','п','р','ш','о','л','д','ь','т','щ'
  db   'з','й','к','ы','е','г','м','ц','ч','н'
  db   'я','х','|','ъ','~',127

; Обработчик прерывания 09h
; Здесь выясняется, нажата ли клавиша-модификатор
; (в данном случае правая клавиша Control)

presence   dw  4376h        ; Ключевое слово присутствия
                            ; драйвера в памяти

int09  label  byte
  cli                       ; Запретить прерывания
  pushf                     ; Сохранить флаги
  push  AX
```

```
        in      AL,060h         ; Прочесть байт из порта 60h
                                ; (клавиатура)
        cmp     AL,0E0h         ; Это байт префикса расширенной
        jne     cont            ; клавиатуры?
        mov     CS:Ext,1        ; Нет, заслать 1 в Ext
        jmp     bye             ; Закончить работу, перейти на
                                ; возврат из прерывания

cont:
        cmp     cs:Ext,1        ; Да, сравнить Ext с 1
        jne     clear           ; Если не равно — перейти к
                                ; очистке Pressed

; Проверка scan кодов на нажатие и освобождение
; правой клавиши Control
        cmp     AL,09Dh         ; Правый Control отпущен?
        jz      break
        cmp     AL,01Dh         ; Правый Control нажат?
        jne     clear

        mov     CS:Pressed,1    ; Заслать 1 в Pressed для проверки
        jmp     cl_ext          ; при следующем вхождении

break:
        cmp     CS:Pressed,1    ; Проверка нажатия и отпускания
        jne     cl_ext          ; правой клавиши Control
                                ; (Pressed равно 1)?
        xor     CS:Cyrillic,Border ; Определение цвета рамки
                                ; экрана

use_border:
        mov     AX,1001h
        push    BX
        mov     BH,CS:Cyrillic  ; Установить цветную рамку
        int     10h             ; для режима вывода кириллицы
        pop     BX

clear:
        mov     CS:Pressed,0    ; Очистить Pressed
cl_ext:
        mov     CS:Ext,0        ; Очистить префикс расширенной
                                ; клавиатуры

bye:
        pop     AX
        popf                    ; Восстановить флаги

db      0EAh                    ; Первый байт команды long jump
old9    label   dword
old9o   dw      0               ; В этих переменных хранится
old9s   dw      0               ; старый адрес прерывания 09h

; Обработчик прерывания 16h
; Здесь производится обработка принимаемых кодов
int16   label   byte
        or      AH,AH           ; Это вход в прерывание на прием
        jz      func0           ; символа через стандартную
                                ; функцию 00h? Если да, то переход
                                ; на func0.
        cmp     AH,10h          ; Это вход в прерывание на прием
        jz      func0           ; символа через дополнительную
                                ; функцию 10h? Если да, то переход
                                ; на func0.
        cmp     AH,1            ; Это вход в прерывание на проверку
        jz      func1           ; наличия и чтения символа через
                                ; стандартную функцию 01h?
                                ; Если да, то переход на func1.
        cmp     AH,11h          ; Это вход в прерывание на проверку
        jz      func1           ; наличия и чтения символа через
                                ; дополнительную функцию 11h?
                                ; Если да, то переход на func1.

db 0EAh                         ; Первый байт команды far jump.
old16   label   dword
old16o  dw  0                   ; В этих двух переменных находи
old16s  dw  0                   ; старый адрес прерывания 16h.

func0:
        pushf                   ; Сохранить флаги
        call    CS:[old16]      ; Вызов прерывания 16h по старому
                                ; адресу
        cmp     CS:Cyrillic,0   ; Сечас используем кириллицу?
        jz      bye16           ; Нет — выход из прерывания
        call    translate       ; Да — перейти к переводу
                                ; принятого кода
bye16:
        iret
func1:
        pushf                   ; Сохранить флаги
        call    CS:[old16]      ; Вызов прерывания 16h по старому
                                ; адресу
        pushf                   ; Сохранить флаги
        cmp     CS:Cyrillic,0   ; Сейчас используем кириллицу?
        jz      by116           ; Нет — выход из прерывания
        call    translate       ; Да — перейти к переводу
                                ; принятого кода
by116:
        popf                    ; Восстановить флаги
        retf    2

; Подпрограмма перевода принятого кода ASCII
; Здесь производится формирование конкретного кода
; в соответствии с раскладкой клавиатуры.
translate proc near
        push    BX              ; Сохранить регистры
        push    DS
        xor     BX,BX           ; Прочесть из BIOS Data Area
        mov     DS,BX           ; байт состояния клавиатуры
        mov     BX,0417h
        test    byte ptr [BX],040h ; Нажат ли какой-либо Shift
                                ; или CapsLock?
        jnz     capslock        ; Да — переход на обработку
                                ; прописных букв
        lea     BX,Ordtbl       ; Нет — подготовить таблицу
                                ; кодировки строчных букв
        jmp     dotrans
capslock:
        lea     BX,Cpstbl       ; Подготовить таблицу прописных
                                ; букв
dotrans:
        or      AH,AH           ; Проверка принятого символа на 0
        jz      notrans         ; Если да, то перекодировка
                                ; не нужна
        cmp     AH,035h         ; Проверка символов на границы
        ja      notrans         ; транслируемой области: если
        cmp     AL,127          ; символ является цифрой, знаком
        ja      notrans         ; препинания или управляющим
        cmp     AL,' '          ; символом, то преобразование не
                                ; требуется.
        jbe     notrans
        add     BL,AL           ; Здесь происходит собственно
        adc     BH,0            ; трансляция кодов символов из
        sub     BX,32           ; первой во вторую половину
                                ; таблицы кодов
        mov     AL,CS:[BX]      ; Занести в AL новый код символа
        xor     AH,AH           ; Очистить AH
notrans:
```

```
    pop     DS              ; Восстановить
    pop     BX              ; регистры
    ret
translate   endp

; Функции и данные, связанные с прерыванием 10h
; (обслуживание видеоадаптера)
mode        db  0
myint10  label  byte
    or      AH,AH           ; Это фунция установки
                            ; видеорежима?
    jz      change_mode     ; Если да, то переход на change_mode
    cmp     AH,11h          ; Это функция вызова
                            ; знакогенератора?
    jz      chargen         ; Если да, то переход на chargen

jmprom:
db 0EAh                     ; Первый байт команды far jump
old10       label  dword
old10o      dw  0           ; В этих двух переменных хранится
old10s      dw  0           ; старый адрес прерывания 10h

change_mode:
    mov     byte ptr CS:mode,AL
    and     AL,7Fh

    cmp     AL,3            ; Проверка на режим 3
    jbe     sett8x14        ; Если < = , то переход на установку
                            ; фонта 8x14
    cmp     AL,7            ; Проверка на режим 7 (моно)
    jz      sett8x14        ; Если да, то переход на установку
                            ; фонта 8x14
    cmp     AL,0Eh          ; Аналогичные проверки на
    jbe     setg8x8         ; графические режимы работы и
                            ; переход на установку фонта 8x8
    cmp     AL,10h          ; Проверка на граф. режим высокого
    jbe     setg8x14        ; разрешения и вызов шрифта 8x14
    mov     AL,CS:mode      ; Восстановить в AL видеорежим
    jmp     jmprom          ; Переход на возврат из прерывания
chargen:
    cmp     AL,30h          ; Это функция получения
                            ; информации о EGA?
    jz      info            ; Если да, то переход на info
    cmp     AL,02           ; Это функция загрузки
                            ; double фонта 8x8?
    jnz     rr              ; Если нет, то переход на rr
    jmp     seta8x8         ; Иначе — переход на seta8x8

rr:
    cmp     AL,23h          ; Это функция установки
                            ; ROM фонта?
    jz      nextchck        ; Если да, то проверять дальше
    cmp     AL,12h          ; Это загрузка ROM фонта?
    jnz     jmprom          ; Если нет, то выход из прерывания
    jmp     sett8x8         ; Иначе — переход на sett8x8

nextchck:
    cmp     BL,3            ; Это установка всех шрифтов 8x8?
    jnz     jmprom          ; Нет — закончить работу
    jmp     seta8x8         ; Да — переход на seta8x8

info:
    cmp     BH,2            ; Получение информации
    je      give8x14        ; о загруженных фонтах
    cmp     BH,3
    je      give8x8
    cmp     BH,4
    je      give8x8top
    jmp     jmprom          ; Завершить работу, выйти из
                            ; прерывания

sett8x14:
    mov     AL,CS:mode      ; Занести в AL видеорежим
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    pushrs                  ; Сохранить регистры
    mov     AX,1100h        ; Подготовка к вызову функции 00h
                            ; знакогенератора — загрузка
                            ; определенного пользователем фонта
                            ; в текстовом режиме.
    push    CS
    pop     ES              ; Заполнение регистров перед
    mov     BP, offset CS:font8x14  ; вызовом прерывания
    mov     CX,256
    xor     DX,DX
    mov     BX,0E00h        ; Число байт на символ в таблице:
                            ; EGA — 0E00h
                            ; VGA — 1000h
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    poprs                   ; Восстановить регистры
    iret

setg8x14:
    jmp     short mysetg
setg8x8:
    jmp     short mysetg8

give8x14:
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    push    CS
    pop     ES              ; Получение в ES:BP адреса
    mov     BP,offset CS:font8x14   ; таблицы символов 8x14
    iret

give8x8:
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    push    CS
    pop     ES
    mov     BP,offset CS:font8x8    ; Получение в ES:BP адреса
    iret                    ; таблицы символов 8x8

give8x8top:
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    push    CS
    pop     ES
    mov     BP,offset CS:font8x8    ; Получение в ES:BP адреса
    add     BP,128*8        ; второй половины таблицы
                            ; символов 8x8
    iret

mysetg:
    mov     AL,CS:mode      ; Занести в AL видеорежим
    pushf                   ; Сохранить флаги
    call    CS:[old10]      ; Вызов прерывания 10h по старому
                            ; адресу
    pushrs                  ; Сохранить регистры
```


5 Зак. 2246

```
    mov   AX,1121h        ; Подготовиться к вызову функции
                          ; 21h знакогенератора — загрузка
                          ; определенного пользователем фонта
                          ; 8x14 в графическом режиме
    push  CS
    pop   ES                     ; Заполнение регистров перед
    mov   BP,offset CS:font8x14  ;вызовом прерывания
    mov   CX,14
    mov   BL,25
    pushf                 ; Сохранить флаги
    call  CS:[old10]      ; Вызов прерывания 10h по старому
                          ; адресу
    poprs                 ; Восстановить регистры
    iret

mysetg8:
    mov   AL,CS:mode      ; Занести в AL видеорежим
    pushf                 ; Сохранить флаги
    call  CS:[old10]      ; Вызов прерывания 10h по старому
                          ; адресу
    pushrs                ; Сохранить регистры
    mov   AX,1121h        ; Подготовиться к вызову функции
                          ; 21h знакогенератора — загрузка
                          ; определенного пользователем фонта
                          ; 8x8 в графическом режиме
    push  CS
    pop   ES                     ; Заполнение регистров перед
    mov   BP,offset CS:font8x8   ; вызовом прерывания
    mov   CX,8
    mov   BL,25
    pushf                 ; Сохранить флаги
    call  CS:[old10]      ; Вызов прерывания 10h по старому
                          ; адресу
    poprs                 ; Восстановить регистры
    iret

seta8x8:
    pushrs                ; Сохранить регистры
    mov   AX,1121h        ; Подготовиться к вызову функции
                          ; 21h знакогенератора — загрузка
                          ; определенного пользователем фонта
                          ; 8x8 в графическом режиме
    push  CS
    pop   ES                     ; Заполнение регистров перед
    mov   BP,offset CS:font8x8   ; вызовом прерывания
    mov   CX,8
    mov   BX,3
    pushf                 ; Сохранить флаги
    call  CS:[old10]      ; Вызов прерывания 10h по старому
                          ; адресу
    poprs                 ; Восстановить регистры
    iret

sett8x8:
    pushrs                ; Сохранить регистры
    mov   AX,1110h        ; Подготовиться к вызову функции
                          ; 10h знакогенератора — загрузка
                          ; определенного пользователем фонта
                          ; в текстовом режиме
    push  CS
    pop   ES                     ; Заполнение регистров перед
    mov   BP,offset CS:font8x8   ; вызовом прерывания
    mov   CX,0100h
    mov   BX,0800h
    xor   DX,DX           ; Очистить DX
    pushf                 ; Сохранить флаги
    call  CS:[old10]      ; Вызов прерывания 10h по старому
                          ; адресу
    poprs                 ; Восстановить регистры
    iret

old1Fo    dw 0            ; В этих двух переменных хранится
old1Fs    dw 0            ; адрес вектора 1Fh

old44o    dw 0            ; В этих двух переменных хранится
old44s    dw 0            ; адрес вектора 44h

; Область хранения данных фонта 8x8
font8x8  label  byte
    INCLUDE 0808.asm      ; В файле 0808.asm хранятся данные
                          ; фонта 8x8

; Область хранения данных фонта 8x14
font8x14  label  byte
    INCLUDE 0814.asm      ; В файле 0814.asm хранятся данные
                          ; фонта 8x14

; Проверка на присутствие драйвера в памяти и начальная
; установка драйвера
INSTALL:
    mov   ax,3509h        ; Получение адреса прерывания 09h
    int   21h
    mov   ax,es:[bx-2]    ; Проверка на присутствие
    cmp   ax,cs:presence  ; программыв памяти
    jnz   load            ; Присутствует — переход на already
    jmp   already

load:
    xor   AX,AX           ; Извлечение из BIOS Data Area
    mov   ES,AX           ; ключевого байта наличия EGA/VGA
    mov   AH,byte ptr ES:[487]
    push  CS
    pop   ES
    and   AH,1000b        ; EGA/VGA присутствует?
    jz    EGA_present     ; Да — переход на EGA_present
    jmp   no_EGA          ; Нет — переход на no_EGA

EGA_present:
    push  ES
    push  DS
    push  ES              ;. Освобождение DOS environment
    mov   AX,DS:[2Ch]     ; для уменьшения занимаемой
    mov   ES,AX           ; драйвером памяти
    mov   AH,49h
    int   21h
    pop   ES

    mov   AX,3509h        ; Получение адреса
    int   21h             ; прерывания 09h
    mov   old9o,BX        ; Сохранение адреса
    mov   old9s,ES        ; прерывания 09h

    mov   AX,3516h        ; Получение адреса
    int   21h             ; прерывания 16h
    mov   old16o,BX       ; Сохранение адреса
    mov   old16s,ES       ; прерывания 16h

    mov   AX,3510h        ; Получение адреса
    int   21h             ; прерывания 10h
    mov   CS:old10o,BX    ; Сохранение адреса
    mov   CS:old10s,ES    ; прерывания 10h

    mov   AX,351Fh        ; Получение адреса
    int   21h             ; вектора 1Fh
    mov   old1Fo,BX       ; Сохранение адреса
    mov   old1Fs,ES       ; вектора 1Fh
```

```
        mov   AX,3544h          ; Получение адреса
        int   21h               ; вектора 44h
        mov   old44o,BX         ; Сохранение адреса
        mov   old44s,ES         ; вектора 44h

        cli                     ; Запретить прерывания
        lea   DX,int09          ; Установить новый
        mov   AX,2509h          ; адрес прерывания 09h
        int   21h

        lea   DX,int16          ; Установить новый
        mov   AX,2516h          ; адрес прерывания 16h
        int   21h

        lea   DX,myint10        ; Установить новый
        mov   AX,2510h          ; адрес прерывания 10h
        int   21h

        lea   DX,font8x14       ; Установить новый
        mov   AX,2544h          ; вектор прерывания 44h — адрес
        int   21h               ; текстового фонта 8x14

        lea   DX,font8x8        ; Установить новый
        add   DX,128*8          ; вектор прерывания 1Fh — адрес
        mov   AX,251Fh          ; графического фонта 8x14.
        int   21h
        sti                     ; Разрешить прерывания

        mov   AX,03h            ; Установить новый видеорежим
        int   10h

        mov   AX,CS
        mov   DS,AX             ; Выдать сообщение
        lea   DX, message       ; об успешной загрузке
        mov   AX,0900h          ; драйвера клавиатуры/экрана
        int   21h

        pop   DS
        pop   ES

        mov   DX, offset Install; Завершить программу,
        int   27h               ; оставив в памяти ее
                                ; резидентную часть

no_EGA:
        mov   AX,CS
        mov   DS,AX
        lea   DX,not_EGA        ; EGA/VGA не присутствует, вывести
        mov   AX,0900h          ; соответствующее сообщение
        int   21h
        jmp   finish            ; Перейти к завершению программы
already:
        mov   AL,DS:[082h]      ; Разбор командной строки из PSP и
        cmp   AL,'/'            ; проверка на ключи выгрузки:
        je    check_key         ; /u
        cmp   AL,'-'            ; /U
        je    check_key         ; /r
        jmp   no_PSP            ; /R
check_key:
        mov   AL,DS:[083h]      ; /k
        cmp   AL,'U'            ; /K
        je    do_unload         ; Вместо символа '/' может
        cmp   AL,'u'            ; использоваться '-'
        je    do_unload
        cmp   AL,'k'
        je    do_unload
        cmp   AL,'K'
        je    do_unload
        cmp   AL,'r'
        je    do_unload
        cmp   AL,'R'
        je    do_unload
        jmp   bad_key

; Блок выгрузки программы из памяти и восстановления
; векторов int 09h, 10h, 16h, 1Fh, 44h
do_unload:
        mov   AX,word ptr ES:old10o ; Получение адресов для
        mov   DX,word ptr ES:old10s ; восстановления
                                    ; вектора int 10h
        mov   CX,ES                 ; В ES сохранено значение
                                    ; CS резидентной части
                                    ; программы
        cli
        xor   BX,BX
        mov   ES,BX
        mov   BX,10h*4
        mov   ES:[BX],AX            ; Восстановление старого
        mov   ES:[BX + 2],DX        ; вектора int 10h из old10

        mov   ES,CX
        mov   AX,word ptr ES:old9o  ; Получение адресов для
        mov   DX,word ptr ES:old9s  ; восстановления
                                    ; вектора int 09h
        xor   BX,BX
        mov   ES,BX
        mov   BX,09h*4
        mov   ES:[BX],AX            ; Восстановление старого
        mov   ES:[BX + 2],DX        ; вектора int 09h из old9

        mov   ES,CX
        mov   AX,word ptr ES:old16o ; Получение адресов для
        mov   DX,word ptr ES:old16s ; восстановления
                                    ; вектора int 16h
        xor   BX,BX
        mov   ES,BX
        mov   BX,16h*4
        mov   ES:[BX],AX            ; Восстановление старого
        mov   ES:[BX + 2],DX        ; вектора int 16h из old16

        mov   ES,CX
        mov   AX,word ptr ES:old1Fo ; Получение адресов для
        mov   DX,word ptr ES:old1Fs ; восстановления
                                    ; вектора int 1Fh
        xor   BX,BX
        mov   ES,BX
        mov   BX,1Fh*4
        mov   ES:[BX],AX            ; Восстановление старого
        mov   ES:[BX + 2],DX        ; вектора int 1Fh из old1F

        mov   ES,CX
        mov   AX,word ptr ES:old44o ; Получение адресов для
        mov   DX,word ptr ES:old44s ; восстановления
                                    ; вектора int 44h
        xor   BX,BX
        mov   ES,BX
        mov   BX,44h*4
        mov   ES:[BX],AX            ; Восстановление старого
        mov   ES:[BX + 2],DX        ; вектора int 44h из old44

        sti
        mov   AH,49h
        mov   ES,CX                 ; Восстановление ES
        int   21h                   ; Освобождение памяти, которую
                                    ; занимала программа
        mov   AX,0003h
```

```
        int     10h
        mov     AX,CS
        mov     DS,AX
        mov     AX,0900h        ; Выдать сообщение об успешной
        lea     DX,unload       ; выгрузке драйвера
        int     21h             ; экрана/клавиатуры
        jmp     finish

no_PSP:
        mov     AX,CS
        mov     DS,AX           ; Драйвер уже присутствует в
        lea     DX,loaded       ; памяти, выдать соответствующее
        mov     AX,0900h        ; сообщение
        int     21h
        jmp     finish

bad_key:
        mov     AX,CS
        mov     DS,AX           ; Неверный ключ запуска.
        lea     DX,err_key      ; Выдать соответствующее
        mov     AX,0900h        ; сообщение
        int     21h

finish:
        mov     AX,4c01h        ; Завершить программу
        int     21h

message db 'Advanced Screen and Keyboard driver for Cyrillic'
        db 0Dh, 0Ah
        db 'Use Right Control'key to select between Latin and'
        db ' Cyrillic code tables', 0Dh, 0Ah
        db 'JVK Software. 1991, June.'
        db 0Dh, 0Ah, '$'
unload  db 'Advanced Keyboard/Screen driver was succefully
        db ' unloaded...', 0Dh, 0Ah, '$'
loaded  db 'Advanced Keyboard/Screen driver was already'
        db ' loaded, use Right Control key...', 0Dh, 0Ah, '$'
not_EGA db 'EGA/VGA is not present', 0Dh, 0Ah, '$'
err_key db 'Invalid key. Available keys for AD_DRV are:'
        db 0Dh, 0Ah
        db '/U or -U', 0Dh, 0Ah
        db '/R or -R', 0Dh, 0Ah
        db '/K or -K', 0Dh, 0Ah
        db 'All the keys are designed for unloading'
        db ' Keyboard/Screen driver from memory.'
        db 0Dh, 0Ah, '$'

ENDS
END     start
```

Ю.Кравацкий

Adobe Systems выпустила Adobe Photoshop 2.0. Программа включает новые режимы для цветоделения, редактирования цветных и черно-белых изображений и импорта файлов в формате Adobe Illustrator.

Программа работает на Macintosh SE или Macintosh II с 2 Мбайтами ОЗУ, жестким диском и операционной системой версий не ранее 6.0.4. Рекомендуется использовать 4 Мбайта ОЗУ, цветной монитор и компьютеры с быстрыми процессорами.

Программа совместима с Apple Macintosh System 7.0. Стоит она 895 долларов, для зарегистрированных пользователей прошлых версий — 149 долларов.

Newsbytes News Network, 17 June 1991

Председатель федеральной резервной системы США, американского центрального банка, заявил, что несмотря на недавнее улучшение экономических показателей, он не видит тенденции к общему улучшению положения дел в американской экономике.

Он сказал, что спад производства несомненно закончился, но это еще не значит, что начался подъем.

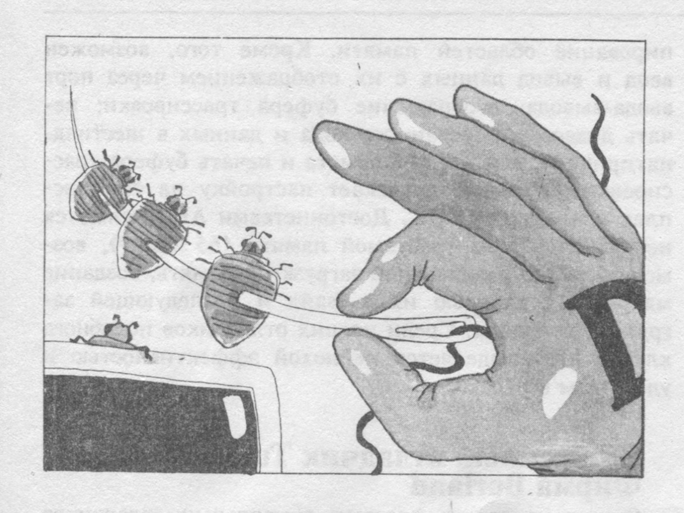
Компьютерные эксперты отмечают, что этот экономический спад сильно, а может быть и непоправимо, затронул компании типа Compaq, продающие свои машины по достаточно высоким ценам. В отличие от предыдущих периодов времени, крупнейшие американские компании стали покупать все больше машин у конкурентов Compaq и IBM, торгующих по более скромным ценам. Более того, они находят, что эти дешевые машины работают ничуть не хуже производимых престижными и дорогими компаниями.

Newsbytes News Network, 18 July 1991

Датчанин Бьйорнер, ведущий специалист по CASE-технологиям и один из крестных отцов языка ADA будет читать в МГУ двухнедельный курс лекций. Представители факультета ВМК МГУ заявляют, что это не первый и не последний случай, когда западных профессоров приглашают читать лекции в Москву.

Newsbytes News Network, June 15, 1991

Фирма GoldStar предлагает советским покупателям модем GMS2400MPC. Модем использует стандартный набор AT-команд, совместимый с Hayes-модемом. Скорость передачи — до 2400 бод в соответствии с протоколами V.22, V.22bis, Bell 212A и Bell 103. Кроме того, модем поддерживает протокол коррекции ошибок и сжатия данных MNP5. Аппарат выполнен в виде стандартной восьмиразрядной платы расширения половинной длины. Цену для советского рынка представитель фирмы сообщить отказался.



Отладчик. Так мы переводим английское слово Debugger. Буквально же оно означает "приспособление для вылавливания блох", в данном случае — в программах. Сегодня отладчик — это инструмент, и инструмент очень мощный. Без его помощи не обходится создание ни одной серьезной программы. Он помогает программисту промоделировать множество ситуаций и, конечно, найти ошибки и устранить недочеты. Но какие бывают отладчики? Ответ вы найдете в этой статье.

Отладчики программ для MS-DOS

С самого начала появления ЭВМ перед программистами со всей остротой встал вопрос отладки разрабатываемых программ. На первых отечественных машинах при программировании в кодах или автокодах (автокоды — это первые ассемблеры, именовавшиеся "один к одному", так как они полностью повторяли программирование в кодах, только в мнемонической записи) этот вопрос разрешался довольно легко, так как присутствовал режим потактового выполнения программ, а на панель ЭВМ в двоичном коде (просто включением лампочек) автоматически выводился адрес выполняемой команды и содержание ячейки памяти по этому адресу.

С появлением первых языков высокого уровня, таких как Algol и FORTRAN, отладку стали выполнять, вставляя различные дополнительные операторы печати, которые позволяли как осуществлять трассировку (слежение за последовательностью выполнения операторов программы), так и получать значения ключевых переменных. Все это имело место при пакетном режиме работы. После появления персональных компьютеров, на которых пользователь работает исключительно в интерактивном режиме, такой способ отладки стал малоэффективен. Это привело к созданию большого числа программ, предназначенных для упрощения процесса отладки и называемых отладчиками (иногда интерактивными отладчиками, что подчеркивало режим их работы).

Отладчиком обычно называют программу, которая позволяет анализировать работу других программ, задавать режимы работы, проверять выполнение различных условий, осуществлять трассировку, контроль и изменение переменных и операторов программы, а также работу с памятью машины и регистрами процессора. Интерактивная отладка является настолько важным средством, что она явилась одним из предметов исследований инициативной группы пользователей LFTF и фирмы IBM. Результаты были изложены в форме общих требований к отладчикам в 1983 г. в работе R. Seidwer и N. Tindall в SOFTWARE ENGINEERING Notes. Отметим, что в последнее время отладчики стали применяться не только для отладки программ, но и для оптимизации некоторых участков программы, нелегального снятия защиты с программ хакерами, определения некоторых внутренних характеристик программ. Для работы в среде MS DOS существует большое число отладчиков с разнообразными характеристиками. Их можно поделить на две боль-

шие группы: отладчики для работы с памятью и программами в кодах ПК и символьные отладчики для работы с программами в исходных кодах языков высокого уровня.

Отладчик DEBUG Фирма Microsoft

Одним из наиболее простых является отладчик DEBUG, который входит в стандартную поставку MS DOS и предназначен для отладки программ на ассемблере, загрузки, изменения и просмотра произвольных файлов, а также выполнения исполняемых файлов. Кроме того, отладчик позволяет работать с текущим содержимым памяти. Если загружаемый файл имеет расширение HEX, то его формат принимается соответствующим шестнадцатиричному формату фирмы INTEL и при загрузке преобразуется в исполняемый формат. Этот отладчик прост в освоении и использовании, однако его возможности довольно ограниченны. Управляется DEBUG командами, каждая из которых задается одной буквой и списком параметров. DEBUG позволяет выводить на экран и изменять содержимое регистров и флагов; просматривать и изменять содержимое как отдельных ячеек памяти, так и целых участков памяти; сравнивать содержимое участков памяти; осуществлять ввод и вывод данных из портов; задавать точки останова; выполнять трассировку программы, а также ассемблирование программы непосредственно в память и дизассемблирование содержимого памяти. Необходимо отметить, что для задания точек останова, трассировки и выполнения программы требуется предварительно настроить соответствующие регистры.

Отладчик AFD

Примерно к тому же классу относится отладчик AFD (автор H.P. Puttkamer). Управление этим отладчиком осуществляется как через меню, так и при помощи командной строки. В главном меню AFD предусмотрены следующие возможности: пошаговый режим без выполнения подпрограмм и с пошаговым выполнением подпрограмм, задание точек останова, поиск в файле, выполнение программ, переход между окнами и получение помощи. Имеется возможность сохранения в файле точек останова с целью их использования при повторном выполнении программы. Отладчик имеет следующие окна: регистров и флагов, адресов и кодов программы, два окна данных и окно задания командной строки. В каждом окне можно работать либо непосредственно с его объектами, либо выполнять операции с помощью команд. Основные команды отладчика позволяют осуществлять следующие операции: загрузить файл в память, начиная с заданного адреса; просмотреть код программы; дизассемблировать программу, начиная с заданного адреса; записать данные в файл; установить значения регистров и флагов; выполнять программу с начала или с текущего адреса; прерывать выполнение по нажатии клавиши; осуществлять поиск данных в памяти, сравнение и копирование областей памяти. Кроме того, возможен ввод и вывод данных с их отображением через порт ввода-вывода; отображение буфера трассировки; печать дизассемблированного кода и данных в шестнадцатиричном или ASCII формате и печать буфера трассировки. Отладчик допускает настройку на тип дисплея и процессора ПК. Достоинствами AFD являются небольшой объем требуемой памяти (65 Кбайт), возможность его резидентной загрузки в память, создание макросов с записью их в файл и последующей загрузкой из файла. Среди прочих отладчиков подобного класса AFD выделяется неплохой эффективностью и удобством в работе.

Встроенный отладчик Turbo Basic Фирма Borland

Одним из самых простых символьных отладчиков является встроенный отладчик системы Turbo Basic фирмы Borland. Данный отладчик позволяет выполнять лишь простейший набор операций при отладке пользовательских программ. Такой неширокий спектр возможностей системы Turbo Basic объясняется тем, что сам язык достаточно прост и ясен.

Для того, чтобы отладить программу, написанную в системе Turbo Basic, необходимо выбрать пункт Debug из главного меню системы. После этого пользователю предоставляется всего две возможности:
- Trace
- Runtime Error.

Первая из этих команд (Trace) позволяет осуществлять трассировку программ. Это стандартное средство, которое обязательно входит в состав любого отладчика. Команда Trace выполняет прогон текущей строки исходного текста программы. Текущая строка при этом выделяется контрастным цветом. При трассировке отображаются имена транслируемых функций и процедур, номер текущей линии и т.п. Для реализации пошагового режима необходимо нажать комбинацию клавиш Alt+F10 (сделать один шаг). Для смены режимов трассировки и выполнения следует использовать клавиши Alt+F9.

Вторая команда Runtime Error, как это видно из ее названия, используется для поиска в исполняемом .EXE файле ошибок библиотеки поддержки. Для того, чтобы найти ошибку, достаточно указать этот пункт меню и, если она выявится в процессе выполнения программы, то будет выдан стандартный номер ошибки и сообщение с разъяснением возникшей ошибки.

Встроенный отладчик Quick Basic Фирма Microsoft

Более мощным отладчиком для языка Basic является встроенный отладчик системы Quick Basic фирмы Microsoft. Работа с ним напоминает работу со встроенным отладчиком системы Quick C, и поэтому пользователю при переходе от одной Quick-системы к другой практически нет необходимости читать руководство и

восполнять свои пробелы в знаниях об этих отладчиках.

Для отладки программы в системе Quick Basic необходимо выбрать пункт Debug в меню, после чего на экране дисплея появится подменю с описанием возможностей отладчика, которые условно можно поделить на 3 основные группы:

- работа с точками останова (т.е. адресами, на которых выполнение программы временно останавливается);
- работа с командами контроля операторов (т.е. спецификациями, описывающими выражения для областей памяти, за которыми ведется наблюдение);
- работа по трассировке программ.

К первой группе относятся следующие команды:

- Toggle Breakpoint;
- Clear All Breakpoint;
- Set next Statement.

С помощью данных команд можно задать точку останова программы, очистить все точки останова, убрать текущую точку останова и т.п. Таким образом, если программа велика и осуществлять ее трассировку по шагам затруднительно (так как это может занять довольно много времени), то помечают точку останова, до которой будет выполняться программа. Далее пользователь может либо просмотреть интересующие его значения переменных, либо продолжить счет в пошаговом режиме, либо выполнять программу до следующей точки останова.

Вторая группа функций при работе со встроенным отладчиком фирмы Microsoft предоставляет пользователю еще более широкие возможности, такие как:

- Add Watch — определить наблюдаемое выражение или область памяти. Значения, описанные этим оператором, выводятся в окно наблюдения. Новые значения добавляются в окно наблюдения каждый раз при изменении текущих значений наблюдаемого оператора в процессе выполнения программы;
- Watchpoint — определяет точку условного останова, которая будет обработана, когда заданное выражение истинно (не нулевое). Наблюдаемый оператор, заданный этой командой, сравнивается в процессе выполнения программы с нулем;
- Delete Watch — удалить указанный наблюдаемый оператор из списка;
- Delete All Watch — удалить все наблюдаемые операторы.

Третья группа функций является довольно стандартной, и поэтому подробно мы ее рассматривать не будем. Отметим лишь, что она позволяет осуществлять трассировку программы и работать в пошаговом режиме.

Таким образом, на примере рассмотренных встроенных отладчиков видно, что, несмотря на их оперативность и удобство работы с пользовательскими программами, спектр их возможностей довольно ограничен, хотя для первой "прикидочной" отладки программы они вполне подходят. К их достоинствам относится то, что они просты в освоении, не требуют много места на диске и в оперативной памяти (в отличие от таких супер-отладчиков, как CodeView фирмы Microsoft и Turbo Debugger фирмы Borland).

Встроенные отладчики Modula-2 и JPI Modula-2 Фирмы Logitech и Jenson & Partners International

Более сложными отладчиками, обладающими гораздо более широкими возможностями, являются встроенные отладчики систем Modula-2 фирмы Logitech и JPI Modula-2 фирмы Jenson & Partners International (причем отладчик второй фирмы заметно мощнее своего конкурента). Отладчик фирмы Logitech предоставляет пользователю стандартные возможности по отладке прикладных программ, такие как постановка и удаление точек останова, использование команд отслеживания операторов, отображение локальных и глобальных переменных и т.п. По своим функциональным возможностям данный отладчик имеет много общего с отладчиком фирмы JPI, но интерфейс общения с пользователем оформлен несколько иначе. В частности, отладчик поддерживает работу с мышью, что существенно облегчает и упрощает общение с ним.

Теперь остановимся более подробно на отладчике фирмы JPI. Он имеет название VID (Visual Interactive Debugger), и для его запуска необходимо выполнить команду:

VID {/опции} «имя .EXE файла»«параметры»,

где опции и параметры задают начальные установки отладчика. Отладчик может быть также запущен из оболочки системы JPI.

Для описания возможностей отладчика VID кратко укажем, что видит пользователь на экране дисплея при отладке своей программы. Основную часть экрана занимает исходный текст программы пользователя, где текущий выполняемый оператор выделен контрастным цветом. В нижней части экрана отображаются команды, с помощью которых пользователь может выполнить программу в указанном режиме (Go menu), указать точки останова и выполнить с ними отдельные операции (Breakpoint menu), произвести трассировку программы (Trace menu) и осуществить ряд других действий. С помощью вызываемых меню можно просмотреть, а в случае необходимости изменить интересующие значения (однако далеко не все). Так с их помощью можно просмотреть содержимое регистров, значения локальных и глобальных переменных, вычисляемые выражения, стек вызова активных процедур, а также просмотреть текст программы в кодах ассемблера, что иногда бывает необходимо для более полного понимания работы программы.

Следует отметить, что последняя версия отладчика VID поддерживает все компиляторы серии Top Speed, а также имеет ряд преимуществ по сравнению с некоторыми другими отладчиками (например, возможность использования расширенной памяти и поддержки раз-

НПО "ПЕРЕДОВЫЕ ТЕХ

Фирма "ТЕХНОТРОНИК" НПО "Передовые технологии"

ПРЕДЛАГАЕТ ЗА ВАЛЮТУ И РУБЛИ



станции компьютерной графики и отдельные комплектующие для телевизионных студий кабельного телевидения на базе компьютеров IBM PC AT 286/386/486.

I. Для студий на основе аппаратуры VHS рекомендуется оборудование, основанное на адаптерах EGA/VGA:

1. EGA→ VIDEO (PAL, UHF) около 500 долл.США
2. EGA→ VIDEO (PAL, GENLOCK) около 1000 долл.США
3. VGA→ VIDEO (PAL, SCART) около 1000 долл.США
4. VGA→ VIDEO (PAL, GENLOCK) около 2000 долл.США
5. VIDEO GRABBER "Screen Machine" (PAL, SECAM) VIDEO→SuperVGA около 2800 долл.США

Используя VGA - VIDEO, Вы сможете получить на видеокассете мультипликации из пакетов ANIMATOR и 3D-STUDIO фирмы AUTODESK, пакетов GRASP, STORYBOARD, слайдшоу фирмы RIX.
На оборудовании типа GENLOCK Вы сможете наложить на уже имеющееся изображение на видеокассете компьютерную мультипликацию, логотип Вашей фирмы, текст. Рекомендуется для студий кабельного телевидения. Возможно использование в качестве системы телетекста.

Когда Вы используете "Screen Machine", полноцветное изображение от одного из трех источников видеосигнала демонстрируется в реальном масштабе времени на экране монитора VGA в окне любого размера. Возможно управление проигрывателем оптических дисков и контроль всех параметров звука и изображения. Поддерживается формат Super-VHS. Применяется для интерактивных информационных и обучающих систем, систем презентации, формирования графических баз данных для различных приложений, тренажеров, для "закачки" видеоизображений в настольных издательских систем с сохранением их на винчестере в стандартных форматах GIF, PCX...

II. Для студий на основе аппаратуры Super-VHS (но без покадрового управления монтажными видеомагнитофонами) рекомендуется оборудование, основанное на адаптерах TARGA/VISTA:

1. TARGA (PLUS 32 PAL) около 4800 дол.США
2. VISTA (4M PAL) около 7000-9000 дол.США в зависимости от комплектации

Обе графические платы являются стандартом de-facto для формирования систем компьютерной видеографики, написания программ мультипликации и даже с точки зрения формата файлов. Оборудование является полноцветным (32 бита на пиксел, то есть более 16,000,000 цветов), имеет видеовход и видеовыход (композитный и S-VIDEO), работает с программой 3D-STUDIO фирмы AUTODESK.
TARGA рекомендуется для использования в программах новостей, спортивных программах. Имеет режимы GENLOCK (наложение графики на видеофон), CROMA KEY ("синий фон" — наложение видеоперсонажа, например, диктора на компьютерную графику) и соответствующее программное обеспечение.
VISTA может быть использована для работы телебиржи. Скоростной графический процессор позволяет в реальном масштабе времени делать сложные эффекты над изображением — типа "перелистывания страниц", что реализовано в некоторых программах, приобретаемых дополнительно. Имеется русский перевод наиболее полезных программных систем.

III. Для студий на основе аппаратуры BETACAM SP (а также VPR, BETACAM, U-MATIC, ONE-INCH, DIGITAL, S-VHS PANASONIC)
рекомендуются профессиональные (broadcast) комплексы компьютерной графики.
Стандартный комплект оборудования состоит из графического адаптера VISTA с платой расширения (14 Мбайт видеопамяти), монитора (20 дюймов), профессиональных декодера (PAL-RGB) и кодера (RGB-PAL), анимационного контроллера (управление двумя видеомагнитофонами), планшета с пером, стандартного компьютера IBM PC 386/486 (10M EMS, 300MB HDD) и транспьютерного ускорителя.
Рекомендуется полный набор программ для моделирования, двухмерной, трехмерной графики, анимации в системе "DSG" фирмы DIGITAL ARTS (частью системы являются программы пакета RENDERMAN фирмы PIXAL, поддерживаются русские фонты формата PostScript, сделан перевод документации). Впрочем, открытая архитектура и стандартное оборудование позволяют использовать программы других фирм, например, AUTODESK или AT&T.

IV. Проводятся консультации по установке и обучение работе на системе 3D-STUDIO фирмы AUTODESK.

Приведенные цены являются приблизительными. Цены на профессиональный комплекс устанавливаются по запросу после составления спецификации. Возможна оплата в рублях. Адаптеры демонстрируются в техническом центре фирмы "Технотроник". Информацию о видеоадаптерах и об условиях поставки Вы сможете получить у Дмитрия Гавриленко или Александра Труханова по телефонам: (095)262-07-65, (095)262-08-33.

личных типов точек останова). Так, если в большинстве отладчиков точки останова обычно подразделяются на простые точки останова (имеют также второе название — "прилипающие" точки останова) и условные точки останова, то в отладчике VID возможно применение шести типов точек останова:

- однократная точка останова (удаляется после прохождения указанного оператора);
- однократная точка останова с задержкой (удаляется после того, как указанный оператор будет выполнен заданное количество раз);
- прилипающая точка останова;
- прилипающая точка останова с задержкой;
- условная точка останова;
- выдача сообщений без прерывания программы.

Отладчик CodeView Фирма Microsoft

Отладчик CodeView фирмы Microsoft является мощным средством отладки программного обеспечения. Он позволяет отлаживать программы как в исходных текстах, так и в ассемблерных кодах. Отладчик универсален в том смысле, что позволяет работать с программами, написанными на различных языках высокого уровня: C, FORTRAN, Pascal, BASIC. Характерной чертой CodeView является необходимость специальной подготовки программ для работы в режиме исходных текстов. Специальная подготовка заключается в компиляции и компоновке программ с использованием определенных опций отладки. Для всех компиляторов и компоновщиков (редакторов связей) фирмы Microsoft эти опции одинаковы.

Если необходимо скомпилировать исходный текст программы, нуждающейся в отладке, то в командной строке компилятора перед именем файла указывают параметр /Zi. Этот параметр означает, что компилятор должен записать номера строк и символьную информацию по именам программы в создаваемый объектный файл. В случае, если программа состоит из нескольких модулей, то совсем необязательно все их компилировать с опцией /Zi для полной отладки. Во-первых, можно вообще не указывать опции отладки, но тогда будет отсутствовать возможность символьной отладки и доступа к локальным переменным тех частей программы, которые содержатся в файлах, скомпилированных подобным образом. Во-вторых, можно скомпилировать части программы с опцией /Zd. При этом в объектный файл будет записана не вся необходимая для отладки информация. Для частей программы, подготовленной таким образом, будет невозможен доступ к внутренним переменным по их символьным именам.

Следует отметить, что все последние версии компиляторов фирмы Microsoft являются оптимизирующими, т.е. результирующие коды программы могут быть определенным образом преобразованы с целью получения наибольшей эффективности, а следовательно, инструкции, находящиеся в программе, могут не соответствовать исходным строкам программы. Поэтому оптимизацию и отладку нельзя использовать совместно. Кроме того, нельзя приступать к отладке программы до тех пор, пока ее компиляция не будет успешно завершена. Отладчик, естественно, не может помочь исправить синтаксические ошибки в исходном тексте программы.

В качестве примера можно привести командную строку для компиляции программы на FORTRAN DEMO.FOR:

```
fl /Zi /c DEMO.FOR.
```

После компиляции всех файлов отлаживаемой программы необходимо выполнить компоновку объектных файлов. Для получения возможности дальнейшей работы с отладчиком редактору связей LINK.EXE фирмы Microsoft необходимо указать параметр /CODEVIEW (он может быть сокращен до /CO). Этот параметр означает, что в рабочий файл необходимо внести адреса символов и строк исходного текста.

Работая с отладчиком, не следует использовать параметр /EXEPACK редактора связей, так как он вызывает удаление всей символьной информации из рабочего файла.

После того как программа подготовлена вышеописанным образом, можно начать ее отладку. Для этого необходимо вызвать отладчик посредством команды CV. Формат этой команды следующий:

CV «опции» имя исполняемого файла «параметры»,

где опции — это один или несколько параметров отладчика. Их указывать необязательно.

Имя исполняемого файла — это рабочий файл отлаживаемой программы. Если указанный файл не имеет нужного формата, т.е. не скомпилирован с опцией отладки, то отладчик начнет работать в режиме ассемблера, выдав при этом предупреждающее сообщение.

Параметры — это строка параметров, необходимых для работы отлаживаемой программы. Для многих программ она необязательна.

Основные параметры отладчика, указываемые в командной строке, имеют следующий смысл:

/B — этот параметр используется в том случае, если необходимо работать в черно-белом режиме монитора;

/S — этот параметр позволяет одновременно работать с графическими и текстовыми изображениями, а также с несколькими страницами видеопамяти, давая возможность постоянно наблюдать выходное изображение. В случае, если отлаживаемая программа работает только в текстовом режиме с одной страницей видеопамяти, отладчик можно запускать с параметром /F. При этом на хранение изображения будет выделено намного меньше оперативной памяти;

/M — этот параметр необходим в том случае, если в конфигурации компьютера присутствует мышь, но ее использование нежелательно (например, при отладке программы управления мышью);

/43 /50 — эти параметры позволяют работать в режимах, предоставляемых адаптерами EGA и VGA,

при которых возможен просмотр 43 (50) строк на экране в текстовом режиме;

/C<...> — этот параметр позволяет сразу после запуска отладчика выполнить одну или несколько его команд. Если команд несколько, то они разделяются знаком ';'.

Управление работой CODEVIEW можно осуществлять двумя способами: посредством меню или функциональных клавиш и через командный режим.

Экран отладчика разбит на несколько окон. Центральное окно содержит текст отлаживаемой программы. В случае, если программа специальным образом подготовлена для отладки, это ее исходный текст, в противном случае — коды ассемблера. В окнах, расположенных в верхней части экрана, содержатся: в левом — локальные переменные программы (только для отладки в исходных текстах), в правом — переменные, за значением которых необходимо наблюдать в процессе отладки. Кроме того, в правой части экрана может располагаться окно с содержимым всех регистров и флагов процессора. Это окно появляется и исчезает при нажатии клавиши F2. В нижней части экрана располагается окно командной строки, в котором можно вводить команды системы. Следует отметить, что в версии 3.0 отладчика появилась возможность видеть и работать сразу с несколькими частями отлаживаемой программы, имея соответствующее количество окон исходных текстов. Переход между окнами исходных текстов и окном команд осуществляется либо с помощью мыши, либо посредством нажатия клавиши F6. При использовании мыши также имеется возможность изменять размеры одного окна за счет другого.

В верхней части экрана располагается ниспадающее меню. Оно содержит следующие пункты:

File — этот пункт позволяет просмотреть какой-либо текстовый файл, завершить работу с отладчиком и временно выйти в операционную систему с возвратом назад по команде exit;

Search — этот пункт позволяет осуществлять поиск текстовых строк и меток в текущем загруженном файле;

View — этот пункт позволяет переключать режимы работы отладчика, меняя принципы отображения исходного текста программы. Таких принципов может быть три:

- режим исходного текста;
- режим ассемблера;
- совместный режим, при котором каждой строке исходного текста ставится в соответствие определенный набор ассемблерных команд, причем отладка идет по ассемблерным командам;

Run — этот пункт позволяет установить программу в начальное состояние, запустить ее на выполнение, включить режим автоматического пошагового выполнения, выполнить один шаг программы;

Options — этот пункт позволяет устанавливать и удалять точки останова, добавлять переменные для контроля и модифицировать их значения;

Calls — этот пункт включает в себя список вызванных на текущий момент подпрограмм с переданными им значениями. В начальном состоянии программы этот пункт ничего не показывает.

Версия 3.0 отладчика позволяет сохранять его состояние и после завершения сеанса работы. Это состояние восстанавливается при последующем запуске отладчика без параметров. Кроме того, в отладчике предусмотрена система контекстно-чувствительной помощи, вызываемая по клавише F1, либо через меню помощи, в котором имеется возможность перемещаться по разделам помощи, как по страницам книги.

Встроенный отладчик Turbo Pascal Фирма Borland

В интерактивной среде Turbo Pascal имеется довольно мощный и удобный отладчик для осуществления отладки на уровне исходного текста. Несмотря на то, что этот отладчик не может обеспечить полного набора средств отладки (например, отладка на уровне машинного кода, вычисление выражений Pascal-программы и т.п.), он позволяет отлаживать подавляющее большинство Pascal-программ. Отладчик Turbo Pascal поддерживает выполнение следующих функций: контроль выполнения программы в пошаговом режиме; отслеживание значений переменных и выражений; изменение значений переменных и элементов структур данных; выполнение и отладку частей программ, невыполняемых при их нормальной работе. Кроме того, имеется возможность контроля стека вызовов, т.е. вывода в окно перечисления всех вызовов процедур и функций, которые в этот момент являются активными. Стек вызовов хранит до 128 вызовов процедур и функций; также при отладке имеется возможность нахождения какой-либо процедуры или функции с целью задания в ней точек останова или списков просматриваемых параметров. Такие действия не влияют на текущее состояние отлаживаемой программы. К несомненным достоинствам отладчика относится возможность отладки оверлейных структур, а также то, что его использование не изменяет размеров программ и не требует вставки в исходный текст дополнительных команд. В процессе отладки возможно производить вычисления в специальном поле для проверки значений переменных и выражений. Перед отладкой необходимо сформировать две таблицы: обозначений и номеров строк. Таблица обозначений — это база данных, в которой содержатся все используемые идентификаторы. Таблица номеров строк служит для установки соответствия между объектным кодом и исходным текстом. Обе таблицы формируются посредством задания директивы компилятора. Если программа содержит модули, которые тоже необходимо отладить, то вначале исходного текста каждого из них надо поместить директиву компилятора. Каковы же ограничения на использование этого отладчика? Конечно, это большие и сложные программы, для отладки которых не хватает памяти. Кроме того,

отладка невозможна в следующих случаях: отсутствует исходный текст программы; процедура или функция является встраиваемой (inline), внешней (external), обрабатывает прерывание, записана в одном из стандартных модулей системы, содержится в модуле, который был скомпилирован с директивой $ D+. Существенным является то, что при отладке нельзя модифицировать целые массивы, записи и файлы, а также нетипизированные параметры, передаваемые в процедуру или функцию.

Для системы Turbo Pascal версии *5.5* и выше в отладчик введены средства объектно-ориентированной отладки. К таким средствам относятся две возможности: пошаговое выполнение программы со слежением за выполнением правил и слежение за объектами данных. Объекты обрабатываются отладчиком так же, как процедуры и записи. Вызовы правил обрабатываются аналогично вызовам процедур или функций. При этом правило воспринимается как единый оператор (без отображения его внутренних операторов), а статистические и виртуальные правила обрабатываются при пошаговой отладке одинаково. Имена правил с префиксом в виде идентификатора объектного типа отображаются в окне стека вызовов. Объекты, аналогично записям, отображаются в окне вычислений. При задании полного имени объекта отображаются только его поля данных. При задании имени правила в окне вычислений отображается ссылочное значение, являющееся адресом кода данного правила. Необходимо отметить, что возможностями, подобными возможностям встроенного отладчика Turbo Pascal, обладают и встроенные отладчики систем Turbo C и Turbo C++.

Отладчик Turbo Debugger Фирма Borland

Мощным отладчиком программ для среды MS DOS является Turbo Debugger фирмы Borland. Он предназначен для обработки EXE-файлов и программ на языках высокого уровня в системах программирования фирмы Borland. Появившись в 1988 г., за последние 3 года этот отладчик претерпел очень существенные изменения, связанные с выходом на сцену объектно-ориентированных языков. Так, вышедшая в 1989 г. версия 1.5 поддерживает отладку объектов в Turbo Pascal версии *5.5*, а последняя — Turbo Debugger 2.0, появившаяся в 1990 г. — в дополнение к этому обеспечивает работу с объектно-ориентированным C++. В этом отладчике реализован мощный механизм прерываний по точкам останова с выражениями и условиями, а также по аппаратным прерываниям. Кроме того, этот отладчик позволяет отлаживать очень большие прикладные программы с использованием виртуальных систем на ПК с процессором 80386 или на двух комплексах, соединенных по последовательному интерфейсу. При использовании этой возможности на удаленном ПК можно отлаживать программы довольно большого размера, так как для работы отладчика здесь требуется всего 15 Кбайт оперативной памяти. Оконная среда и активные клавиши совпадают с интегрированной средой и активными клавишами Turbo Pascal и Turbo C. При объектно-ориентированной отладке отладчик обеспечивает просмотр классов и инспекцию классов и объектов. Также возможно применение обратных вычислений для упрощения многократных инструкций. К очень привлекательным особенностям этого отладчика относятся такие возможности, как отладка резидентных программ и драйверов устройств, построение ассемблерных программ, отладка регистров процессора 80386, возможность прохода через структуры и связанные списки при отладке данных, а также просмотра стеков вызовов и предыстории вычислений. Отладчик поддерживает работу с мышью. Для работы с объектами в Turbo Debugger предусмотрены новые окна: окно иерархии объектов и окно проверки объектного типа. Первое из них предназначено для просмотра иерархии объектов. В нем отображается информация об объектных типах, а не об экземплярах объектов. В двух подокнах этого окна соответственно перечисляются объектные типы (в алфавитном порядке), используемые в отлаживаемом модуле, и положение объектных типов в иерархической структуре. Во втором окне (проверки объектного типа) дается обобщенная информация по объектному типу, не связанная с конкретным экземпляром этого типа. Оно также состоит из двух горизонтальных подокон, в которые выводятся перечисления полей данных объектного типа и имена правил, а если правило является функцией, то и тип возвращаемого значения. Кроме того, в окне Module можно использовать подокно проверки экземпляра объекта. Это окно предназначено для получения информации о данных конкретного экземпляра объекта.

Семейство отладчиков Periscope 1 Фирма Periscope

Еще одним мощным отладочным средством является семейство отладчиков Periscope 1 (2, 3, 4) одноименной компании. Periscope 1 представляет собой программно-аппаратный комплекс, включающий в себя плату памяти объемом 512 Кбайт (с возможностью расширения до 1 Мбайта), защищенную от записи средствами MS DOS, систему управления прерываниями и программное обеспечение. Стоимость всего комплекса 795 долл. Применение специальной платы позволяет полностью избежать использования основного ОЗУ для отладки. Вместе с тем это достоинство не идет в ущерб надежности хранения отладочной информации. Periscop 1 предоставляет не менее широкие возможности, чем полностью реализованная программная версия Periscop 4. Periscop 2 включает управление прерываниями и программное обеспечение стоимостью 175 долл. Periscop 3 включает плату с 64 Кбайт оперативной памяти для записи собственного программного обеспечения и отладочной информации, аппаратный контроль точек прерывания, буфер для трассировки в реальном времени, что позволяет находить ошибки, которые при использовании програм-

мных средств ищутся очень долго или вообще пропускаются. Для машин с тактовой частотой до 10 МГц стоимость комплекса Periscop 3 составляет 1395 долл. Если в его состав входит новая модель платы, то стоимость увеличивается до 1995 долл.

Необходимо подчеркнуть, что аппаратная поддержка добавляет новые качества, необходимые для решения действительно сложных задач отладки. Аппаратный контроль прерываний позволяет осуществить прерывание в любой физический момент времени. При этом возможна немедленная трассировка ошибок или только их контроль без перезагрузки. Возможен также откат назад.

Для использования Periscop необходимо следующее оборудование: IBM PC XT, AT, PS/2 или 80386 (Periscop 3 требует как программной, так и аппаратной совместимости, он не работает на PS/2 и системе 80386), DOS версии старше 2.0, 64 Кбайта ОЗУ для работы (при установке необходимо 128 Кбайтов), один дисковод и монитор, поддерживающий в текстовом режиме 80 символов в строке.

Программные средства Periscop позволяют отлаживать все виды программ (драйверы устройств, резидентные программы, программы, управляемые прерываниями, и даже не DOS-программы) и обеспечивает поддержку исходных кодов программ на языках высокого уровня и ассемблере. В Periscop 4 реализованы уникальные возможности, которые трудно, а некоторые невозможно найти у других отладчиков. Наиболее интересные среди них — это просмотр локальных символов из MS C (версия старше 5.0); отладка оконных приложений; установка контрольных точек в оверлейных программах; вывод переменных монитора в окно слежения; поддержка при отладке регистров системы 80386; перекодировка символов и дизассемблирование; отладка с использованием терминала ввода-вывода; смешанная символьная отладка; задание контрольных точек для значений флагов.

Другие средства отладки

Новым отладочным средством является Turbo Profiler фирмы Borland, по утверждению фирмы — первая в мире интерактивная информационная программа для программирования в среде MS DOS. Хотя она и не является собственно отладчиком, ее применение будет безусловно способствовать как процессу отладки, так и проектированию более эффективной программы. Какую же информацию можно получить при ее помощи? Во-первых, это различные временные параметры программы: частота вызовов подпрограмм и модулей, время работы каждой подпрограммы. Во-вторых, параметры программы, относящиеся к ее расположению в памяти, использование файлов, последовательность вычислений. Кроме того, отслеживаются характеристики организации оверлеев (с точки зрения их эффективности), работы прерываний. Имеется возможность строить профиль программы на уровне инструкций процессора для программ с соответствующим исходным кодом. Программа снабжена удобным интерфейсом пользователя с системой меню, поддержкой мыши, многократным перекрытием окон и прокруткой изображений. Для многих параметров возможна выдача статистики в форме гистограмм. Эта утилита работает со следующими языками фирмы Borland: TASM 1.0, 2.0; Turbo C 2.0; Turbo C++; Turbo Pascal 5.0.

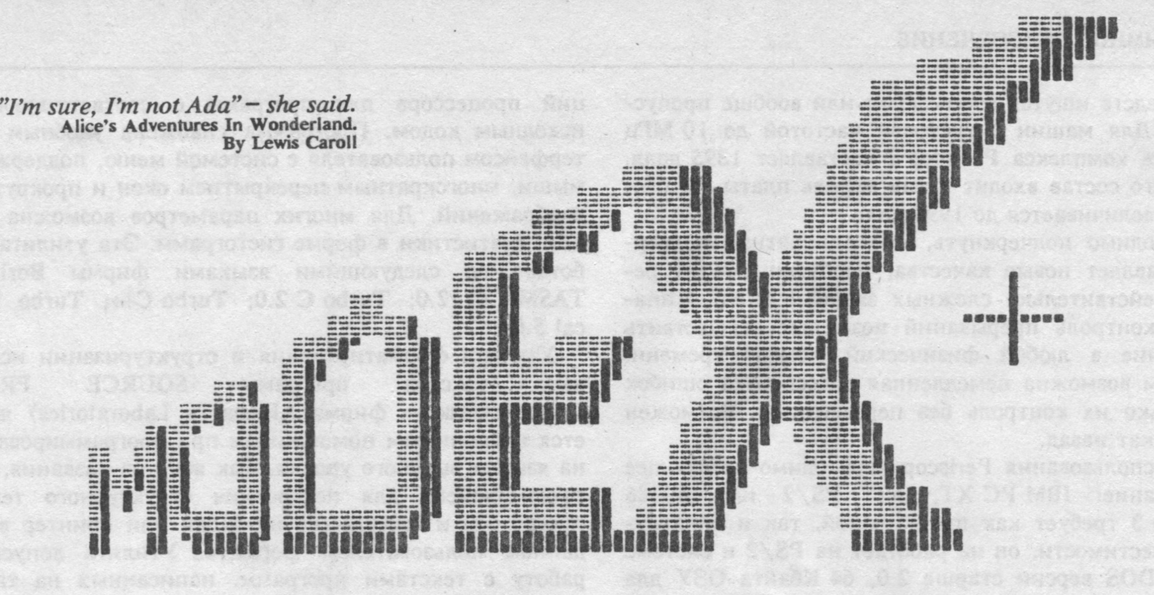
Утилита форматирования и структуризации исходных текстов программ SOURCE PRINT (разработчик — фирма Aldebaran Laboratories) является неоценимым помощником при программировании на языках высокого уровня. Как ясно из названия, она предназначена для построения структурного текста программы и выдачи его на экран или принтер в заданном пользователем формате. Утилита допускает работу с текстами программ, написанных на таких языках, как BASIC, Modula, Pascal, C и языке СУБД dBASE. В состав утилиты входят файлы SP.EXE, MS.EXE, SPBAT.BAT и файлы с демонстрационными программами DEMO.C, DEMO.PAS, DEMO.BAS, DEMO.PRG, DEMO.MOD. Объем, занимаемый утилитой на жестком диске, около 350 Кбайт. Управление работой утилиты осуществляется из командной строки, либо с помощью меню. В первом случае командная строка имеет вид:

SP <имена файлов>/<опции>.

Для вызова меню необходимо запустить программу MS.EXE. При работе с меню пользователь может получить дополнительную информацию об утилите, используя клавишу F1. Кроме того, находясь в меню и производя выбор требуемых условий структурирования и печати, можно сохранить соответствующие опции в файле SPBAT.BAT. Первоначально файл SPBAT.BAT содержит настройку для вывода на принтер структурного текста программы на языке Pascal из файла DEMO.PAS. Помимо построения структуры, утилита обеспечивает следующие возможности: задание интервалов между строками, соединение частей программы из разных файлов в одну программу, построение таблицы перекрестных ссылок, блокировку синтаксического анализа строк, восстановление того вида программы, который она имела до структуризации, выделение в тексте программы ключевых слов, нумерацию строк, изменение вида признака конца файла, управление шириной страниц, установку левого отступа текста, выбор директории и программы, задание количества строк на странице, управление принтером и некоторые другие.

А.Смородинский, А.Воскресенский, С.Ансилевский

Использованы материалы:
BYTE, Jan, 1989
PC World, Jan, 1989, Turbo Assembler & Debuger



MODEX-Plus — система плюс наша поддержка

MODEX-Plus — профессиональная система программирования, ориентированная на промышленную разработку сложных программных комплексов, в первую очередь систем реального времени. Система основана на языке Modula+2 — расширении языка Modula-2.

MODEX-Plus функционирует на IBM PC (XT, AT) в среде MS-DOS в качестве кросс-системы и на ЭВМ СМ-1420, Электроника-79, Электроника-85, ДВК-3, ДВК-4 и других ЭВМ, совместимых с PDP-11, под управлением операционных систем RT-11, РАФОС, TSX-Plus, NTS, RSX-11M, ОС РВ. Сгенерированный код может выполняться на любых машинах с системой команд PDP-11 (включая процессоры серий 1801, 1806, 588).

ВОЗМОЖНОСТИ СИСТЕМЫ

- создание загрузочных комплексов для выполнения в автономном (без операционной системы) режиме, а также под управлением RT-11, РАФОС, TSX-Plus, NTS, RSX-11M, ОС РВ;
- раздельное размещение кодов и данных для прошивки программ в ПЗУ; язык описания неоднородной памяти целевой ЭВМ;
- использование расширенной памяти для создания комплексов, превышающих 64 Кбайта (с возможностью глобальной оптимизации оверлейной структуры);
- унифицированный интерфейс с другими системами программирования; сопряжение с Си и языком Ассемблера, встроенный Ассемблер;
- компактный и эффективный набор расширений языка Modula-2: фиксированная точка, кодовые вставки, структурные константы, инициализация переменных;
- хороший результирующий код за счет использования оптимизирующих транслятора и компоновщика.

Система включает в себя компилятор, компоновщик, мэйкер, эмулятор целевой ЭВМ, динамический и статический (посмертный) символьные (в терминах исходного языка) отладчики, Ассемблер, библиотеку модулей (ввод-вывод, окна, строки и др.), вспомогательные утилиты, интегрирующие оболочку с развитой системой контекстной подсказки (на русском языке) и экранный редактор. Система снабжена комплектом подробной документации.

ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ:

- поставка программно-аппаратного комплекса на базе IBM PC/AT, включающего аппаратный эмулятор целевой ЭВМ; комплекс позволяет производить отладку программ под управлением операционных систем RT-11, TSX-Plus, RSX-11M;
- организация обучения работе с MODEX-Plus;
- доработка MODEX-Plus в соответствии с конкретными условиями применения.

Поставка машинных носителей с MODEX-Plus означает не завершение, а лишь начало сотрудничества МП "МОДЕКС" Поставка гарантирует:
- помощь в установке системы;
- консультации;
- устранение замеченных ошибок и недостатков;
- поставка новых версий с 50% скидкой;
- возможность приобретения новых продуктов по сниженной цене;
- любые иные формы сотрудничества, разрешающие возможные проблемы эксплуатации MODEX-Plus.

Продолжающееся развитие MODEX-Plus включает в себя разработку кодогенераторов для i286 (MS-DOS, QNX) и VAX-11 (VMS, ULTRIX), стенда программно-аппаратной отладки бортовых вычислительных комплексов. Ведутся исследования принципиально новых, нетрадиционных инструментальных средств.

МП "МОДЕКС"обладает опытом создания систем программирования на базе языка Modula+2 для специализированных ЭВМ с нестандартной системой команд и готово рассмотреть предложения на разработку подобных систем.

Телефоны: (095)234-00-33 доб.505 Сабуров, Ермаков (с 11 до 17)
(095)329-94-84 Михаил Сабуров (с 18 до 21)
(095)115-63-07

Данная статья перепечатывается нами с любезного разрешения фирмы Nantucket из первого номера журнала Nantucket News на русском языке, распространяемого в СССР среди зарегистрированных пользователей пакета Clipper. Распространение этого журнала — лишь один из примеров широкого круга услуг, оказываемых этой фирмой зарегистрированным пользователям своего программного обеспечения.

Что такое препроцессор Clipper 5.0?

В книге А. и D.Ullman "Principles of Compiler Desiqn" дается следующее определение препроцессоров:

Препроцессоры создают входной текст для компиляторов и могут выполнять следующие функции:

1. Обработка макроопределений.
2. Включение файлов.
3. "Рациональная" предобработка.
4. Расширение языка.

Препроцессор Clipper предназначен для выполнения всех этих функций.

За пределами dBase макроопределения — это текстовые строки, которые используются как сокращения длинных выражений. Очень часто эти сокращения позволяют увеличить удобочитаемость исходного текста. Мы будем называть эти сокращения псевдофункциями во избежание терминологической путаницы.

Возможность включения файлов в текст программы повышает уровень модульности программирования. Она позволяет хранить исходную программу в нескольких файлах, разбивая ее по определенному принципу, например по назначению, применяемому аппаратному обеспечению, программистам или рабочим группам.

Рациональная предобработка — это процесс расширения старых языков с целью создания более совершенных структур управления, структур данных или синтаксиса языка. По всей вероятности, программисты будут использовать Clipper-препроцессор в основном не для того, чтобы расширять старый язык, а для дополнения этого языка новыми возможностями.

Возможность расширения языка знакома Clipper-программистам по функциям пользователя. Препроцессор Clipper 5.0 позволяет программистам выполнять любые модификации грамматики и синтаксиса языка, что на несколько порядков выше простого определения функций.

Препроцессор Clipper

Clipper всегда имел препроцессор в составе компилятора, который вызывался во время работы CLIPPER.EXE. Отличие Clipper 5.0 состоит в том, что теперь программист имеет к нему доступ. Использование Clipper-препроцессора отличается от традиционных реализаций препроцессоров тем, что он не может выполняться отдельно. Препроцессор в версии 5.0 все также вызывается во время работы CLIPPER.EXE. А доступ к нему теперь

осуществляется одним из четырех способов: посредством задания переключателей в командной строке компилятора, посредством директив препроцессора в тексте исходной программы, при обращении к стандартному файлу описаний Clipper STD.CH, а также всеми названными методами одновременно.

Переключатели в командной строке компилятора

В командной строке могут использоваться следующие переключатели:

/D«идентификатор»[= «текст»] — вызывает замену в тексте программы идентификатора на текст

/P[«имя файла»] — вызывает размещение обработанного препроцессором текста в файле с соответствующим именем

/I«путь доступа» — вызывает включение препроцессором пути доступа в список поиска файлов, заданный переменной среды INCLUDE

/U«альтернативный файл описаний» — вызывает использование препроцессором альтернативного файла заголовков (HeaderFile) вместо STD.CH.

Ниже приводится пример командной строки вызова компилятора:

```
CLIPPER MyProg /DNUMVAL=99 /PMyProg /Ic:\include /UMySTD.CH
```

В этой командной строке происходит компиляция файла MYPROG.PRG под управлением следующих переключателей:

/DNUMVAL — влияет на препроцессор так же, как и строка программы:

```
#define NUMVAL 99
```

/PMyProg — указывает, что следует создать файл MYPROG.PPO и записать в него текст, полученный после препроцессорной обработки. Использование переключателя /P вместе с переключателем /S позволяет выполнить двухшаговую компиляцию программы. Текст после работы препроцессора будет записан в MYPROG.PPO, а затем он может быть откомпилирован.

/Ic:\include — указывает, что препроцессору в первую очередь следует просматривать директорию c:\include при поиске файлов, заданных директивой #include.

/UMySTD.CH — приводит к замещению стандартного файла заголовков Clipper STD.CH на файл MYSTD.CH. Этот переключатель позволяет полностью изменить язык Clipper, но если вы все же решитесь его использовать, то будьте крайне осторожны. Хотя этот элемент Clipper и обеспечивает предельную программную гибкость, он определенно не для малоопытных и трусливых.

Итак, мы рассмотрели первый способ доступа к препроцессору — применение переключателей в командной строке компилятора. Этот способ имеет глобальную сферу действия, то есть его результаты распространяются на весь текст откомпилированной программы.

Директивы в исходном тексте программы

Директивы препроцессора — это команды компилятору, расположенные не в командной строке, а в самом тексте программы. Они используются для того, чтобы изменить способ обработки компилятором некоторых идентификаторов или частей программы. Директивы позволяют "научить" компилятор новым командам и конструкциям языка или даже заставить его "забыть" то, что он уже умеет. Несмотря на то что директивы могут располагаться в любом фрагменте исходного текста, сфера их влияния будет распространяться лишь на тот модуль, в котором они заданы. Другими словами, любые изменения в языке, будут иметь место только в том случае, если компилятор считает текст, содержащий информацию о том, как осуществлять эти изменения.

Ниже мы остановимся на директивах препроцессора более подробно.

Стандартный файл заголовков

Стандартный файл заголовков Clipper — это своего рода "рецепт", который препроцессор должен прочитать перед тем, как компилятор сможет "приготовить" исходный текст для создания объектного кода. В нем хранятся инструкции для обработки языка Clipper. Препроцессор выполняет два действия: синтаксический разбор и перевод. Синтаксический разбор состоит в распознавании частей текста программы как правильных выражений языка.

Затем эти выражения переводятся в вызовы функций и команд, воспринимаемых компилятором. Начиная с версии 5.0, язык Clipper существует как двухкомпонентная языковая система, включающая "метаязык" высокого уровня и "примитивный" язык низкого уровня. Препроцессор считывает хорошо знакомые функции и команды, скажем DBEDIT() или INDEX ON, и переводит их в "примитивные" эквиваленты, которые и записываются в промежуточный файл. Можно сказать, что примитивная компонента языка является собственно языком Clipper, на который препроцессор может перевести синтаксис любого "мета-языка".

А нужно ли все-таки использовать препроцессор? Совсем не обязательно

Теперь, когда вы приблизительно представили себе возможности препроцессора, может возникнуть вполне резонный вопрос: зачем вообще его программировать? Следует сказать, что это совсем не обязательно. Можно продолжать создавать прикладные программы, как вы это всегда и делали, и препроцессор будет ра-

ботать так же, как и всегда. Использование стандартного языка, определенного в стандартном файле описаний, гарантирует, что ваши программы будут понятны любому Clipper-программисту. С другой стороны, допустимо, что программист может определить такой синтаксис, который никому не будет понятен, кроме него, в некотором роде персональный язык программирования. По мере того как число написанных вами на Clipper 5.0 программ будет расти, вы обнаружите, что во многих ситуациях препроцессор в состоянии сэкономить ваши время и силы.

Директивы

Если вы все же решили управлять языком программирования, то для начала я бы порекомендовал вам обратиться к директивам препроцессора. Они позволяют программировать язык, предоставляют доступ к его внутренним возможностям, позволяют упростить программы и облегчают работу с ними. В оставшейся части этой статьи приводятся примеры использования и детально описываются директивы препроцессора #command и #translate.

#command/#translate

Пара #command/#translate — это сердце и душа препроцессора. Обе эти директивы заставляют препроцессор построчно обрабатывать исходный текст программы и формировать выходной текст. Ниже описываются возможные компоненты исходных и результирующих шаблонов, используемых в этих директивах.

«Исходный шаблон»

Литеральные строки

Литеральные строки — это алфавитно-числовые последовательности символов, которые должны быть заменены входным текстом. Они соответствующим образом преобразуются препроцессором и транслируются. Давайте рассмотрим пример, взятый из стандартного файла STD.CH. Если исходная программа содержит строку:

```
CLS
```

то препроцессор заменит ее согласно директиве, определенной в STD.CH:

```
#command CLS = »__Clear()
```

Ключевые слова и идентификаторы

Ключевые слова и идентификаторы — это элементы исходного текста программы, которые должны соответствовать обычным соглашениям языка dBase — отсутствие различий строчных и прописных букв, обязательность первых четырех символов, кроме того, в начале «исходного шаблона» должно стоять какое-либо слово. Это последнее условие является неким искусственным приемом, ограниченность которого лучше всего продемонстрировать на примере. Давайте введем простую команду, присваивающую значение переменной:

```
/* #command is shown to keep it simple
#command «anIdentifier» put: «value»;
= » ; «anIdentifier» : = «value»

cMyStuff put: "this is a test"
? cMyStuff

wait
return
```

Реакция препроцессора на этот код будет следующей:

```
Loading standard defs from STD.CH ...24600 bytes for 264
standard rules. Copiling PP3222.PRG (C50A073) PP3222.PRG:
Error, line 3:[2013] Chunk marker appears only on right side of
#translate or #command 1 error

No code generated
```

Если изменить определение так, чтобы в начале «исходного шаблона» стояло слово, то обработка пройдет успешно.

```
#command at: «anIdentifier» ;
put:«aValue» = » «anIdentifier» : = «aValue»

at: cMyStuff: put "this is test"
? cMyStuff

wait
return
```

Исходные маркеры

Исходные маркеры указываются в тексте программы, а их формат определяет последующий способ их обработки препроцессором. Разные формы исходных маркеров позволяют отразить все возможности существующего языка Clipper и в то же самое время обеспечить достаточную гибкость в реализации его новых черт. Ниже описываются существующие типы исходных маркеров.

«idMarker»

Регулярный исходный маркер является простейшим из входных маркеров. Он уже использовался в примере предыдущего раздела и предназначен для замещения любого правильного входного выражения.

«idMarker,...»

Исходный маркер-перечень в некотором роде более интересен. Кроме замещения единичных правильных выражений, он также позволяет замещать списки, содержащие несколько выражений, разделенных запятыми. Этот исходный маркер используется в случае команд и функций с переменным числом аргументов. В приводимом ниже примере мы рассмотрим результат работы препроцессора, выдаваемый при использовании переключателя компилятора /P. Файл,

полученный после обработки препроцессором исходного текста, иллюстрирует результат обработки исходного маркера-перечня.

Эта программа получает в качестве аргумента список символьных строк и выводит их на экран:

```
#command at: «anObject» put: «value» ;
= » «anObject»:= «value»

#command tell: «aList,...» ;
printYourself = » QOut(«aList»)

at: cHueyStuff put: "Huey"
at: cDueyStuff put: "Duey"
at: cLueyStuff put: "Luey"
at: cDuckStuff put: "Donald"

tell: cHueyStuff, cDueyStuff, ;
  cLueyStuff printYourself
tell: cDuckSTuff printYourself && non-lists work too

wait
return
```

Результат работы препроцессора, который компилятор сформирует в виде файла .PPO, будет выглядеть так:

```
cHueyStuff: = "Huey"
cDueyStuff: = "Duey"
cLueyStuff: = "Luey"
cDuckSTuff: = "Donald"

QOut(cHueyStuff,cDueyStuff,cLueyStuff)
QOut(cDuckStuff)

__Wait()
return
```

Конструкция tell:«alist,...» printYourself аналогична определению команды "?" в STD.CH, и, так же как и команда"?", она переводится в вызов функции QOut().

«idmarker: wordlist»

Исходный маркер с ограничением замещает текст на одно из слов, перечисленных в списке wordlist. Слова в списке разделены запятыми. Такая форма маркера полезна в том случае, когда необходимо ограничить спектр правильных символов, задаваемых в команде. Любое слово, не входящее в этот список, воспринимается как синтаксическая ошибка. Рассмотрим следующий пример:

```
#command SayHey «salut: hello, ;
bonjour, ojala, yo» ;
= » Qout («salut»)
SayHey hello
SayHey bonjour
SayHey ojala
SayHey yo
SayHey hiya

return
```

Препроцессору такой текст не понравится, и он выведет сообщение об ошибке:

```
Loading standard defs from STD.CH...
24539 bytes for 264 standart rules.
Compiling RESTRICT.PRG RESTRICT.PRG:
Error, line 9: 1005 Illegal statement:
SAYHEY HIYA 1 error
No code generated
```

Результат обработки текста препроцессором позволяет точно определить неверную команду:

```
Qout( hello )
Qout( bonjour )
Qout( ojala )
Qout( yo )
SayHey hiya

return
```

Обратите внимание, что шаблон "SayHey hiya" остался необработанным, так как в директиве отсутствует соответствующий образец для его замещения.

«*idmarker*»

Произвольные исходные маркеры замещают любой текст, начиная с позиции, в которой располагается сам маркер, и вплоть до конца оператора. Иными словами, они оказывают влияние на все, следующие за ними, символы команды. Они используются тогда, когда необходимо сформировать командную строку, не соответствующую синтаксису dBase. Пример произвольного исходного маркера взят также из STD.CH.

Конструкция IF...ENDIF обрабатывается до слова "ENDIF", и, следовательно, игнорируются все стоящие за ним символы. Такой способ обработки директивы несет в себе ряд полезных аспектов, связанных с документированием текста программы:

```
#command ENDIF «*x*» = » endif

nLevel : = 0
cOutString0 : = "level 0"
cOutString1 : = "level 1"
cOutString2 : = "level 2"
cOutString3 : = "level 3"

IF nLevel = 0
 ? cOutString0
 IF nLevel = 1
   ? cOutString1
   IF nLevel = 2
    ? cOutString2
     IF cOutString = 3
       ? cOutString3
     ENDIF level 3.
   ENDIF level 2
 ENDIF level 1
ENDIF level 0

RETURN
```

Текст, полученный после обработки препроцессором, будет иметь следующий вид:

```
nLevel : = 0 cOutString0 : = "level 0"
cOutString1 : = "level 1"
cOutString2 : = "level 2"
cOutString3 : = "level 3"

IF nLevel = 0
  QOut( cOutString0 )
  IF nLevel = 1
    QOut( cOutString1 )
    IF nLevel = 2
```

```
      QOut( cOutString )
      IF cOutString = 3
        QOut( 0cOutString3 )
      endif
    endif
  endif
endif

RETURN
```

Обратите внимание, что текст, следующий за ENDIF, удален.

Необязательные предложения

Необязательные предложения определяют те элементы исходных шаблонов, которые могут присутствовать, а могут и отсутствовать в исходной командной строке. Они заключаются в квадратные скобки [] и могут быть вложенными друг в друга.

```
//Here is the COPY TO definition taken directly from STD.H
but indented here for clarity. Note that ALL is the default state so
no result patterns are required for this clause

#command COPY [TO «file»];
        [DELIMITED [WITH «*delim*»] ;
        [FIELDS «fields,...»] ;
        [FOR «for»] ;
        [WHILE «while»] ;
        [NEXT «next»] ;
        [RECORD «rec»] ;
        [«rest:REST»] ;
        [ALL] ;
=» ;

      __dbCopyDelim( «(file)», ;
      «(delim)», ;
      { «(fields)» }, ;
      «{for}», ;
      «{while}», ;
      «next», ;
      «rec», ;
      «.rest.» )

USE Sales
COPY TO TmpSales FIELDS Account,
Balance FOR Region = "SW"
RETURN
```

После препроцессорной обработки текст будет выглядеть следующим образом:

```
if (.F.) ;  __dbSelect(0) ; end;
  __dbUse( "Sales",, if(.F. .OR. F., ! ;
.F., NIL), .F.) ; if(.F.);
  __dbSetIndex({ });
end
  __dbCopyDelim( "TmpSales",,{"Account",;
"Balance"},{|| Region = "SW"},,,, .F.

RETURN
```

Да, ничего не понятно! Давайте разбираться.

```
// is there an area in use?
if (.F.) ; __dbSelect(0) ; end ;

//use the file

  __dbUse( "Sales",, if(.F..OR..F.,! ;
.F., NIL) ,.F.) ;

//we didn´t specify an index
if (.F.) ; __dbSetIndex( { } ); end

//now the COPY

__dbCopyDelim("TmpSales",;//«file»
                ,;//«*delim*» not specified
{"Account","Balance"},;//«fields,..»
{|| Region = "SW"}   ,;//«for»
                  ,;//«while» not specified
                  ,;//«next» not specified
                   ;//«rec» not specified
            .F. )  //«rest:REST» default

RETURN
```

Теперь все это по крайней мере можно прочесть. Как вы видите, перевод препроцессора стал достаточно понятен. Выражения FIELDS и FOR переведены и размещены в соответствующих местах. Пустыми строками помечены предложения, не определенные в командной строке.

«Результирующий шаблон»

Литеральные строки

Литеральные строки прямо переносятся в выходной текст в неизменном виде без синтаксического разбора или вычисления.

Слова

К словам относятся ключи и правильно построенные идентификаторы. Они также передаются в неизменном виде.

Результирующие маркеры

«idmarker»

Регулярный результирующий маркер является простейшим в этой группе. Он просто повторяет на выходе текст, замещающий «idmarker». Если текст, замещающий маркер, отсутствует, то ничего не записывается. Следующий простой пример иллюстрирует замещение текста правильным регулярным результирующим маркером.

```
// Here is simple example
#command SET TYPEAHEAD TO «n» = » ;
SetTypeahead( «n» )
- our regular result marker

SET TYPEAHEAD TO 64 RETURN
```

После препроцессора получается:

```
SetTypeahead( 64 )
- result

RETURN
```

Регулярный результирующий маркер передает значение в выходной текст неизмененным.

#«idmarker»

Строковый результирующий маркер заключает замещающее входное значение в кавычки и переписывает его в выходной текст. Если значение в команде отсутствует, то на выход передается пустая

строка. В следующем примере рассматриваемый результирующий маркер замещает значение, описываемое исходным маркером-перечнем.

```
#command PARSE «vars,...» = » ;
StrParse( #«vars» )

PARSE cMemVar1, cMemVar2, cMemVar3

RETURN
```

Мы видим, что после препроцессорной обработки перечень заключается в кавычки.

```
StrParse( "cMemVar1, cMemVar2,;
cMemVar3" )

RETURN
```

«"idmarker"»

Обычный результирующий строковый маркер отличается от предыдущего тем, что если входной текст отсутствует, то в выходном тексте не размещается пустая строка. Следующий пример содержит по сравнению с предыдущим небольшое изменение.

```
#command PARSE «cVar» = » ;
StrParse( «"cVar"») 

PARSE cMemVar1
RETURN
```

Результат работы препроцессора:

```
StrParse( "cMemVar1" )

RETURN
```

«(idmarker)»

Анализирующий строковый маркер-результат замещает входной текст только в том случае, если он не является сложным выражением. Его аналитическая способность проявляется только в способности различать простые и сложные выражения, что и демонстрирует приводимый ниже пример.

```
#command SET PRINTER TO «file» = » ;
Set( _SET_PRINTFILE, «(file)» 

SET PRINTER TO OUTFILE

SET PRINTER TO (outfile)

RETURN
```

Обратите внимание на разницу в том, как препроцессор обработал эти две эти команды:

```
Set( 21, "OUTFILE" )

Set( 21, (outfile) )

RETURN
```

«{idmarker}»

Блоковый результирующий маркер записывает входной текст, замещающий маркер, в выходной текст в виде блока кода.

Если отсутствует входной текст, то в выходной текст ничего не записывается. Следующий пример взят из STD.CH.

```
#command SET FILTER TO «xpr» = » ;
__dbSetFilter( «{xpr}», «"xpr"» )

SET FILTER TO Cost » 99.99

RETURN
```

Препроцессор формирует следующий текст:

```
__dbSetFilter( {|| Cost » 99.99}, ;
"Cost » 99.99" )

RETURN
```

«.idmarker.»

Логический результирующий маркер записывает в выходной текст .T., если присутствует какой-либо замещаемый текст, и .F. в противном случае. Он полезен при выполнении каких-либо логических вычислений во время компиляции, что и демонстрируется в следующем примере.

```
#command SET MESSAGE TO «n» [«cent: ;
CENTER, CENTRE»] ; = » ;
Set( _SET_MESSAGE, «n» ) ;
Set( _SET_MCENTER, «.cent.» )

SET MESSAGE TO 23
SET MESSAGE TO 23 CENTER

RETURN
```

Результат препроцессорной обработки показывает, что происходит в зависимости от наличия (или отсутствия) опции CENTER:

```
Set( 33, 23 ) ;
Set( 34,.F. )
Set( 33, 23 ) ;
Set( 34,.T. )

RETURN
```

Повторяющиеся предложения

Повторяющиеся предложения указывают препроцессору, что следует повторить предложение в выходном тексте для каждой текстовой цепочки на входе по всем результирующим маркерам, входящим в предложение. Следующий пример повторяющегося предложения показывает, как выполняется выделение и повторение входных символов:

```
#command STORE «value» TO «var1» ;
[,«varN»] = » ;
«var1»: = [ «varN»: = ] «value»

STORE 42 TO nVar0
STORE 42 TO nVar1, nVar2, nVar3
RETURN
```

Можно заметить, что результирующии шаблон "[«varN»: =]" наложен на ",nvar2,nvar3", что позволило препроцессору использовать одно и то же предложение при обработке нескольких входных текстовых цепочек:

```
nVar0: = 42
nVar1: = nVar2: = nVar3: = 42

RETURN
```

Другими словами вместо "nVar2: = " препроцессор создал выражение "nVar2: = nVar3: = ".

Заключение

Итак, вы знаете теперь, как воспользоваться препроцессором. Он даст вам и силу, и гибкость в использовании его весьма широких возможностей. Но помните, что препроцессор все же создан, чтобы облегчить вам жизнь. Не стоит использовать его, если это приведет к усложнению, а не упрощению ваших программ.

Эд Белл,
директор отдела
технического обслуживания Nantucket Corporation

По материалам:

A. and J.Ullman. Principles of Compiler Design, Addison-Wesley.

A.Schreiner and H.Frierman. Introduction to Compiler Construction with UNIX, Jr., Prentice-Hall.

R.Hunter. The Design and Construction of Compilers, John Wiley&Sons.

Если вы желаете получить дополнительную информацию о том, как стать зарегистрированным пользователем Clipper, а также о размещении в Советском Союзе дилерской сети фирмы, обращайтесь в официальное представительство фирмы Nantucket — СП "Магнит" по адресу 127018 Москва, 2-я Ямская ул., 15. Тел. 289-44-77, 289-44-83.

U.S. Robotics купила Touchbase Systems, фирму, производящую карманные модемы WorldPort для переносных машин. В настоящее время Touch-Base является единственным производителем портативного модема, работающего по протоколу V.32 со скоростью 9600 бит в секунду. Условия покупки не разглашались, так как обе фирмы являются частными компаниями, не выпускающими акций.

Председатель U.S. Robotics Кейси Ковелл (Casey Cowell) отметил, что это третье приобретение фирмы за последние 2 года. В 1990 году USR купила производителя коммуникационной программы Blast, фирму Communications Research Group, а в 1989 году — Miracom Technology, лидера на рынке модемов в Великобритании.

В результате сделки TouchBase станет подразделением портативных продуктов фирмы U.S. Robotics. Разработка и производство серии WorldPort будет, тем не менее, продолжаться.

Как сказал сооснователь TouchBase Майк Бернард, "пришло время и нем присоединиться к семье U.S. Robotics." Частью сделки является то, что г-н Бернард станет вице-президентом U.S. Robotics, опекающим отдел портативных продуктов.

Представитель официального дистрибьютора TouchBase в Британии сказал, что сделка поможет расширить круг продаваемый модемов TouchBase. "Но это будет достаточно сложно, потому что модемы TouchBase хороши и сами по себе — фирма имеет около 50 дистрибьюторов в различных странах мира".

Newsbytes News Network,
May 17, 1991

Clipper 5.0
заговорил по-русски

Учитывая потребности рынка, фирма Nantucket подготовила специальную версию Clipper 5.0. Теперь Clipper заговорил по-русски.

Уже создана русская документация, включая и встроенную (Norton Guide), готов русский интерфейс всех утилит (DBU, PE, RL, DOT, отладчик), устранена дискриминация кириллицы при обработке символьных строк (функции UPPER(), LOWER(), ISUPPER(), ISLOWER(), ISALPHA() теперь распознают буквы кириллицы точно так же, как и латинские). Кроме того, теперь кириллицу воспринимают шаблоны "A" и "N", функция "@A" предложения PICTURE команды GET и функция "@!" команды SAY; команда SORT и функция ASORT(). Функции CMONTH() и CDOW() возвращают названия месяцев и дней недели по-русски.

Сообщения об ошибках времени выполнения заносятся во внутреннюю экспортируемую переменную err:description ERROR-объекта на русском языке. Разумеется, все комментарии в исходных текстах программ (.PRD) и файлов описаний (.ch) во всех поддиректориях на русском языке.

Несомненно, что став более доступной и понятной, русская версия Clipper 5.0 будет пользоваться еще большей популярностью среди наших программистов.

Графический интерфейс и распространение идей СУБД на область графики

Очень часто пользователи сталкиваются с необходимостью создания интерактивных графических сред, и оказывается, что использование обычных графических пакетов типа HALO88, библиотек работы с форматом PCX и т.п. создает массу проблем и заставляет работать с графикой на низком уровне, а использование "монстров" типа Autocad само по себе задача не из простых. До появления представляемой графической системы не существовало простого и технологичного средства для разработки диалоговых графических систем, так необходимых при создании современных комплексов программ, тренажеров, обучающих и графических информационных систем. Кроме того, совершенно открытой оставалась область работы с технологическими и функциональными схемами для которой CAD-системы слишком велики, а обычные графические пакеты слишком малы.

В широко распространенных языках программирования, таких как C, Pascal, Basic и других включен набор функций элементарной (базовой) графики. В этот набор входят функции рисования простых геометрических фигур: линий, прямоугольников, многоугольников, окружностей, эллипсов, дуг и т.д. Кроме того, набор дополняется функциями закраски областей, выдачи строк текста, средствами рисования графиков и диаграмм, функциями для работы со спрайтами. Существуют также отдельные библиотеки с расширенным набором низкоуровневых графических функций, что важно для таких языков, как CLIPPER, не имеющих собственных средств поддержки графики. Средства элементарной (базовой) графики рассчитаны на область деловой графики, т.е. построение на экране всевозможных графиков и диаграмм, а также на показ простых схем, рисунков и картинок (слайдов), подготовленных в графических редакторах. Однако есть целый класс задач, в которых графика не только служит для показа на экране результатов расчета или готовых форм, но и является интерфейсом компьютера с пользователем. На современном уровне компьютеризации свойство интерактивности присуще практически всем программам для ПК в текстовом режиме. Интерактивность в графике дает качественно более высокий уровень взаимодействия с пользователем. Достаточно сказать, что простое указание курсором на экране в графическом режиме может заменить продолжительное листание справочников, как это обычно делается в текстовом режиме. Объем информации на одном экране в графическом режиме также может быть значительно больше, чем на том же экране в текстовом режиме.

К области интерактивной графики относится целый ряд задач, таких как разработка:

- диспетчерских систем,
- систем для работы с картами и схемами,
- тренажеров, обучающих систем,
- рекламной графики,
- мультипликации, компьютерных игр.

При создании программ и систем в этих областях программист сталкивается с большими трудностями, если использует средства только базовой графики. Возникающие проблемы решаются при использовании специально разработанного графического интерфейса.

Описываемый графический интерфейс состоит из набора библиотек функций базовой и интерактивной графики и графического редактора. Библиотеки разработаны для языков программирования: C, Turbo-Pascal, Clipper.

Графический редактор и библиотеки взаимосвязаны концептуально, то есть графический редактор построен с использованием библиотек графического интерфейса, а библиотеки включают функции для работы с графическими объектами, подготовленными при помощи графического редактора, — слайдами, томами, фреймами. Например, построив карту или схему тренажера в редакторе, программист может затем использовать их в программе, при этом программа независима от конкретного расположения элементов на экране. Карта или схема может быть скорректирована в редакторе, — изменены цвет и расположение элементов без изменения программы.

Графический интерфейс оперирует тремя типами графических объектов:

Слайд — растровая копия экрана — хранится в файле с расширением *.DMP или *.PCX в упакованном формате. Создается в графическом редакторе или импортируется из других систем при помощи резидентного копировщика экрана. Формат PCX — стандартный и применяется во многих графических системах. Слайд обычно используется в качестве фоновой картинки.

Том — библиотечный набор картинок небольшого формата. Доступ к картинкам осуществляется по номеру или по имени. Картинки хранятся в томе упакованном формате. Тома удобны для хранения примитивов (icons), элементов меню, условных обозначений для карт, фаз движения для мультипликации, часто используемых фрагментов изображения.

Фрейм — графическая база данных — состоит из упорядоченного набора записей и заголовка. Запись содержит информацию о типе, координатах, цвете и других атрибутах одного из графических элементов. Графический элемент это — линия, прямоугольник, окружность, многоугольник, закраска, сплайн, текстовая строка, картинка из тома. В заголовке содержится информация о слайде и томе, которые используются с этим фреймом. Запись может быть дополнена кодом, определяемым пользователем, тогда каждому элементу изображения будет поставлен в соответствие код, что можно использовать, например, для связи с базой данных и организации доступа к элементам изображения.

Предоставляемый набор функций графического интерфейса достаточен для написания собственного графического редактора с нужными вам свойствами. Функции в библиотеке разделены на две группы:

- функции базовой графики,
- функции интерактивной графики.

Набор функций базовой графики представляет собой стандартный набор функций рисования простых графических элементов, функций вывода текстовой информации, форматированного ввода, работы с окнами, переключения страниц, скроллинга экрана, показа слайдов, вывода на принтер.

Набор функций интерактивной графики позволяет работать с фреймами, слайдами и томами, а также с курсором через манипулятор "мышь" или клавиатуру. Функции для работы с фреймами построены по принципам, похожим на те, которые заложены в идеологию языка dBase. Поэтому интерфейс легко вписывается в СУБД CLIPPER.

В состав функций работы с фреймами входят следующие функции:

- чтение одного или нескольких фреймов в память,
- актуализация одного из фреймов (переключение текущего фрейма),
- показ всех элементов фрейма на экране,
- показ выбранного (текущего) элемента фрейма на экране,
- установка текущим первого или последнего элемента фрейма,
- последовательный просмотр элементов фрейма,
- изменение атрибутов элемента фрейма (координат, цвета),
- выбор элемента фрейма через указание его на экране,
- присоединение дополнительного кода к элементу фрейма,
- выбор элемента фрейма по коду,
- запись измененного фрейма в файл,
- удаление фрейма из памяти.

В качестве примера работы с несколькими фреймами можно привести задачу, использующую туристическую схему города, где один фрейм отражает план города, второй фрейм — транспортную сеть, третий — расположение гостиниц, кинотеатров, ресторанов и т.д. Таким образом создается многослойная карта, и программирование такой задачи значительно упрощается. Такую карту можно создать или скорректировать с помощью графического редактора. Использование фреймов снимает с разработчика трудоемкую часть работы с графикой и позволяет уделить больше внимания логике самой задачи.

В группу функций работы с томами входят следующие функции:

- загрузка оглавления тома в память,
- загрузка картинки из тома по имени или порядковому номеру,
- показ картинки на экране в разных режимах, например, часто используется режим наложения на фон с просветом на месте точек с нулевым цветом.

Входящий в описываемую графическую систему редактор поддерживает векторный и растровый способы создания изображения. По возможностям растровой графики редактор стоит на уровне таких графических систем, как Paint Brush, Dr.Halo, Sb Plus, и др. Наличие шрифтов с кириллицей делает этот редактор более привлекательным для отечественного пользователя, а наличие векторной графики выводит его на качественно новый уровень. Часто задают вопрос — чем была вызвана необходимость разработки нового графическо-

го редактора? Дело в том, что существующие редакторы работают или с растровой, или с векторной графикой, а предпринимавшиеся попытки объединения этих двух направлений (например, в пакете Fanta Vision) были очень робкими и страдали незавершенностью. Мы решили объединить оба направления в одну систему, добавив к этому некоторые идеи работы с графическими объектами и представление векторной картинки как базы данных. Здесь очень существенным, на наш взгляд, представляется именно распространение некоторых идей и подходов СУБД к графике. Мы определяем для графических баз данных операции, присущие обычным базам, что дает совершенно новый подход к работе с компьютерной графикой. Этот подход отличает предельная простота при работе с графическими объектами и элементами и отсутствие каких-либо искусственных интеллектуальных надстроек, которые обычно называют "философскими основами продукта", но на самом деле втискивают пользователя в "прокрустово ложе" идей разработчика.

Графический редактор поддерживает следующие режимы мониторов:

16 (0x10) — EGA HiRes (640x350, 16 цветов)
14 (0x0E) — EGA MeRes (640x200, 16 цветов)
13 (0x0D) — EGA LoRes (320x200, 16 цветов)
18 (0x12) — VGA (640x480, 16 цветов)
113 (0x71) — VGA71 (800x600, 16 цветов)

По умолчанию инициализируется режим EGA HiRes. Работа осуществляется мышью или с клавиатуры. Графический редактор предоставляет следующие возможности:

- рисование линий, причем линии могут быть достаточно сложными (например, волнистыми), а простые типы линий могут иметь разную толщину;
- рисование точками разного размера — это чисто растровое рисование, служащее для корректировки и создания слайдов. При этом доступны режимы рисования через точку или четными или только нечетными точками, а также рисование точкой с заданным шаблоном закраски;
- заливка, перекраска, обмен двух цветов; особенно интересно то, что точка инициализации закраски является элементом фрейма. Следовательно, этот элемент можно удалить и заменить на другой, что весьма непросто в других редакторах при сложных шаблонах закраски;
- рисование прямоугольников, окружностей, замкнутых многоугольников;
- рисование сплайнов, дуг (гипербол, парабол, эллипсов), ломаных линий;
- ввод текстовой информации различными шрифтами, вертикально или горизонтально с масштабированием и тенями;
- вертикальный скроллинг экрана, позволяющий использовать всю видеопамять;
- печать выбранной области экрана на матричном принтере;
- рисование в режиме увеличения (zoom);
- копирование части растрового экрана;
- поворот выбранного элемента или области растрового экрана на 180 градусов относительно вертикальной или горизонтальной оси.

Перечисленные режимы достаточно стандартны и практически не отличаются от подобных режимов других редакторов. Гораздо более интересны режимы работы с графическими элементами:

- при работе с томом растровых картинок очень легко листать этот том, брать нужную картинку и расставлять ее в любых позициях экрана, при этом вы сочетаете возможности растровой и векторной графики — сама растровая картинка хранится в томе и ее использование в нескольких местах экрана практически не требует дискового пространства, фрейм увеличивается всего лишь на десятки байт. Не составляет проблем взять новую картинку с экрана и поместить ее в том, откорректировать уже существующую в томе картинку, удалить картинки из тома или поменять их местами. Более того, нет необходимости каждый раз перемещать курсор к тому, любую картинку можно выбрать "не сходя с места", так как нажав правую клавишу мыши (Esc), вы переходите в режим выбора картинки. Каждый том — это отдельный файл, хранящий растровые картинки, количество и размер которых ограничены только величиной свободной оперативной памяти при работе с редактором, обычно эта величина порядка 200 Кбайт. При загрузке фрейма и/или использовании шрифтов размер свободной оперативной памяти несколько уменьшится, но не намного, так как эти объекты не занимают больших объемов памяти;
- в редакторе существует возможность просмотра фаз движения растровых картинок из тома в режиме мультипликации. При этом вы определяете, насколько согласованы между собой фазы движения, а несложная программа позволит вам получить на экране движущийся объект;
- векторное (не включающее в себя растровых элементов) изображение можно масштабировать, уменьшая или увеличивая в любой пропорции;
- элементы можно отмечать, удалять с экрана, сдвигать, поворачивать, изменять их атрибуты (например, поменять цвет многоугольника или изменить шаблон закраски области), причем некоторые операции можно проводить как с отдельными элементами, так и с группами элементов. Каждый элемент может иметь код, и из своих программ вы можете работать с этими кодами.

Таким образом, вы видите, что постепенно мы подходим к идее, что пользователь может создавать свои программы интерактивной графики; рисуя элементы на экране и работая с их кодами из программ — можно полностью абстрагироваться от графики и работать только с кодами. Это упрощает программирование, так как задачу распознавания элемента мы берем на себя. С помощью такого подхода можно просто рисовать графические оболочки программ, тренажеры, системы графических меню и т.п.

Коды можно хранить в базах данных, связывая через них графические элементы с любой информацией о них. Это естественным образом приводит к созданию библиотек для системы Clipper, поскольку перечисленные идеи и возможности распространяют подход СУБД на область графики. При этом необходимо еще раз отметить, что мы рассматриваем фрейм как базу графических данных, на которой определены следующие действия: встать на первый элемент фрейма, перейти к следующему или предыдущему элементу, определить текущий элемент и количество элементов во фрейме, нарисовать текущий элемент, изменить код текущего элемента, сохранить измененный фрейм. Можно даже работать с несколькими фреймами одновременно, устанавливая один из них в качестве активного. Наличие даже такого небольшого базового набора операций позволяет создавать сложные программы причем с минимальными усилиями.

Для среды Clipper написаны также функция просмотра и корректировки базы данных в графическом режиме (подобие DbEdit) и функция перевода текстового экрана в графический, что позволяет органично связать Clipper и графические библиотеки в единую систему, а не просто подставить к системе Clipper дополнительные графические возможности.

Особое внимание уделяется корректной работе с оперативной памятью: при необходимости можно выгрузить ненужные шрифты, фреймы и тома. Это очень существенный момент, так как графика требует наличия больших объемов оперативной памяти.

Рассмотрим примеры типичных программ на языке Clipper, написанные с использованием данного подхода и позволяющие судить о технологичности, естественности и удобстве программирования. Приведенные примеры являются готовыми интерактивными средами, а маленький логический шаг к пониманию того, как получить графическое меню, мы предлагаем вам сделать самостоятельно.

Пример программы, показывающей тип элемента, на котором находился курсор при нажатии правой клавиши мыши.

```
InitGraph(16)  && инициализация графического
               && режима EGAHi
InitMouse()    && инициализация мыши
aa = LoadFrame('test') && загрузка фрейма в память
showframe(aa)  && показ фрейма на экране
ch = 0
do while ch<>27
 ch = Mmouse() && чтение нажатий мыши или клавиатуры
 if ch = 13
 c = get_obj() && получение адреса элемента под курсором
 b = gett_obj(c) && получение типа элемента по его адресу
 mark_obj(c,15) && отметка указанного элемента 15-м цветом
 do case
 case b = 'L'
 msg = 'линия'
 case b = 'B'
 msg = 'прямоугольник'
 case b = 'C'
 msg = 'окружность'
 case b = 'G'
 msg = 'сплайн'
 case b = 'U'
 msg = 'дуга'
 case b = 'S'
 msg = 'текст'
 case b = 'I'
 msg = 'картинка'
 case b = 'F'
 msg = 'закраска'
 case b = '1'
 msg = 'ошибка чтения'
 otherwise
 msg = 'нет элемента
 endcase
 gsay(30,20,msg)  && выдача на экран сообщения
                  && об элементе под курсором
 endif
enddo
 ginkey(0)        && задержка до нажатия клавиши мыши
                  && или клавиатуры
delframe(aa)      && выгрузка фрейма из памяти
CloseGraph()      && возвращение к текстовому режиму
```

Пример программы, показывающей код элемента, на котором стоит курсор при нажатии левой клавиши мыши или клавиши Enter клавиатуры.

```
InitGraph(16)  && инициализация графического
               && режима EGAHi
InitMouse()    && инициализация мыши
aa = LoadFrame('test') && загрузка фрейма в память
showframe(aa)  && показ фрейма на экране
msg = 'XXXXXXX'
do while msg<>space(7) && цикл, пока не нажата
                       && правая клавиша мыши
msg = getgcode()  && получение кода объекта под курсором
gsay(30,20,msg)   && выдача на экран кода элемента
                  && под курсором
enddo
delframe(aa)      && выгрузка фрейма из памяти
CloseGraph()      && возвращение к текстовому режиму
```

Пример программы, определяющей количество окружностей во фрейме.

```
InitGraph(16)       && инициализация графического
                    && режима EGAHi
aa = LoadFrame('test') && загрузка фрейма
                    && с именем "TEST.FRM"
last = Last_Obj()   && определение адреса
                    && последнего элемента
addr = First_Obj()  && определение адреса первого
                    && элемента фрейма
sum = 0
do while addr<= last && цикл, пока адрес не превышает
                    && адрес последнего элемента фрейма
 if gett_obj(addr) = 'C' && определение типа элемента
```

```
                    && по его адресу
 sum = sum + 1
 endif
addr = Next_obj(addr)  && получение адреса
                    && следующего элемента
enddo
gsay(30,20,'Количество окружностей: ' + str(sum,3))
ginkey(0)           && задержка до нажатия клавиши мыши
                    && или клавиатуры
delframe(aa)        && выгрузка фрейма из памяти
CloseGraph()        && возвращение к текстовому режиму
```

Эти же программы будет выглядеть аналогично, будучи написанными на языках Pascal и Си, так как сохраняются идеология и имена функций библиотек.

В состав поставляемой графической системы также входят:
- резидентный копировщик экранов,
- программа просмотра графических объектов (фреймов, слайдов, и томов),
- конвертер фреймов в исходные тексты программ — очень интересное средство, позволяющее буквально рисовать некоторые части программ.

В качестве примера приведем текст программы, полученной из фрейма "TEST.FRM", который был использован в тестовых программах. Текст дается в статью без каких-либо изменений. Комментарии генерируются автоматически вместе с текстом программы.

```
Set Escap Off
Set Talk Off
Set Bell Off
Set Scoreboard Off
Set cursor Off
Clear All
Set Conf On
Set Scoreboard Off
Set Cursor Off
Set Wrap On
Declare MPPX[255],MPPY[255] && координаты
                    && для многоугольника и сплайна
InitGraph(16)       && Инициализация графического
                    && режима EGAHi
 * где : EGALo = 13 { 320x200, 16 цветов }
 * EGAMe = 14 { 640x200, 16 цветов }
 * EGAHi = 16 { 640x350, 16 цветов }
 * VGA = 18 { 640x480, 16 цветов }
 * VGA71 = 113 { 800x600, 16 цветов }
InitMouse()         && Инициализация мыши
 SetActPage(0)      && активна нулевая страница
 GSlid("ROCK")      && загрузка слайда
chu = LoadVol("HORSE2.VOL") && загр. оглавл. тома
                    && с картинками
                    && загрузка картинок
setgcolor(15)       && задание цвета
SetLnStyle (1,0)    && задание типа линии
Line(89,65,39,204)  && рисование прямой
SetFillPat(0,1)     && задание типа закраски — простой цвет
Bar(30,249,70,329)  && закрашенный прямоугольник
SetLnStyle (0,0)    && тип линии — простой
Rectangle(30,249,70,329) && прямоугольник
FillCircle(106,148,40) && закрашенный круг
Circle(77,110,2)    && окружность
SetLnStyle (1,0)    && тип линии
DrCurve( 0,162,291,105,360,161,387) && рисование дуги
Line(211,518,198,378) && рисование прямой
MPPY[1] = 46
MPPX[1] = 209
MPPY[2] = 103
MPPX[2] = 170
MPPY[3] = 182
MPPX[3] = 230
MPPY[4] = 7
MPPX[4] = 171
NPP = 4
FillPoly(NPP, MPPX,MPPY) && закраска многоугольника
DrawPoly(NPP,MPPX,MPPY) && рисование ломаной линии
Line(171,7,209,46)  && рисование прямой
SetGFnt( 3, 0, 1, 1) && загрузка шрифта
snf = 'text'        && строка текста
DrText( 68,386,snf) && написание текста
SetFillPat(0,3,1)   && задание типа закраски — образцом
Paint(219,29)       && заливка ограниченной области
SetFillPat(1,3,15)  && задание типа закраски — образцом
Paint(187,36)       && заливка ограниченной области
PutImgS ( 86,444, 1) && отображение картинки из тома
PutImgS (267,328, 3) && отображение картинки из тома
ginkey(0)           && задержка до нажатия клавиши мыши
                    && или клавиатуры
CloseGraph()        && возвращение к текстовому режиму
```

Рассмотренная в данной статье система была представлена на заседаниях Московского и Ленинградского Clipper-клубов, где получила высокую оценку специалистов.

За информацией о системе можно обращаться в представительство фирмы Nantucket в Москве к Дмитрию Фивейскому (тел. 289-44-77) или к авторам статьи в Тверь (тел. [08222]-204-32).

Илья Биллиг
Леонид Цыпунов
Геннадий Юдин



Среди прочих программ упаковки данных специализированные архиваторы исполняемых файлов занимают особое место. Ведь при разработке методов упаковки именно исполняемых файлов порой рождаются самые удивительные идеи, которые находят свое воплощение в невиданных ранее фантастических программах. Об одной из таких программ — архиваторе LZEXE — и пойдет речь в этой статье.

ПРОГРАММЫ УПАКОВКИ ДАННЫХ

Архиватор LZEXE

Тот, кто постоянно работает с компьютером, знает, что никогда нельзя считать себя слишком богатым и полностью обеспеченным оперативной памятью и дисковым пространством... Во времена CP/M все программы были жестко ограничены 64 Кбайтами ОЗУ. Сюда входили и такие "монстры", как WordStar, dBASE, SuperCalc. Большинство программ писались на 8080-ассемблере, и даже самые "крутые" из них занимали не более 45 Кбайтов дискового пространства. И многим пользователям щекотали нервы те "гигантские" объемы дискового пространства, которыми обладала их система, располагающая двумя флоппи-дисководами по 160 Кбайтов каждый.

Затем в бой вступили IBM PC с их 640 Кбайтами ОЗУ и жесткими дисками объемом (страшно подумать!) аж по 10 Мбайтов. Сегодня размеры жестких дисков подошли к гигабайтным отметкам, а на серверах глобальных сетей и десятками гигабайтов уже никого не удивишь. Но ведь нам уже и этого мало! (Вот она, ненасытная человеческая сущность!) Один мой приятель, активный пользователь BBS и сети RELCOM, работает на 80386/25 МГц машине с 4 Мбайтами ОЗУ и 300 Мбайтным жестким диском, и регулярно, примерно раз в месяц, оказывается в том "интересном" положении, когда жесткий диск забит "под завязку" и приходится архивировать все, что можно, чтобы спастись от этого информационного наводнения. В чем тут дело? Мне думается, что здесь мы сталкиваемся с компьютерным воплощением известного принципа "природа не терпит пустоты". Это значит, что сколько бы мы ни наращивали объемы своих жестких дисков, наши программы в ответ будут из сил выбиваться, но постараются все это заполнить.

Файл-серверы изначально были задуманы, как спасение от наших с вами дисковых проблем. Да, сначала их объемы выглядели, как Гулливер в стране лиллипутов — локальных дисков пользователей. Вдобавок они позволяли пользователям сети иметь на всех лишь одну — общую копию каждой программы (включая все ее оверлейные и вспомогательные файлы) и работать с этой программой на любом компьютере сети. Однако и по сей день пользователи сетей упорно продолжают поиски до-

полнительных дисковых возможностей для своих компьютеров.

В чем же тут дело? По-видимому, часть проблемы заключена в нас самих, и виной тому наша невообразимая жадность. Мы, подобно Плюшкину, собираем и собираем нужные и ненужные программы, архивируем их, складываем в "ящик", и затем снова собираем. Нам невыносима мысль об отправке какого-нибудь файла на "силиконовые небеса" из страха, что однажды нам понадобится именно этот 250 Кбайтный .EXE-файл, который к тому времени уже два года "провалялся" без дела.

Другой неиссякаемый источник пополнения наших дисков — неэффективные компиляторы языков программирования, с помощью которых создаются многие современные программы. Только представьте себе, что Clipper транслирует однострочную программу вывода одного лишь слова на экран в исполняемый файл размером около 150 Кбайтов!

Способы борьбы с растущими и множащимися файлами известны давно. Альтернативой покупке все новых и новых дисков служит архивация файлов. В предыдущих номерах КомпьютерПресс мы уже рассказывали об утилитах-архиваторах, о программах резервного копирования, о "невидимых" архиваторах, осуществляющих сжатие и распаковку файлов в фоновом режиме. Сегодня речь пойдет о сжатии .EXE-файлов и об оригинальном архиваторе LZEXE.

Само по себе сжатие .EXE-файлов не представляет какой-то особой проблемы. Любые архиваторы (за исключением специальных, предназначенных для сжатия лишь определенных типов файлов) превосходно справляются с этой задачей. Но при этом практически у всех архиваторов имеются те или иные недостатки (о достоинствах архиваторов более подробно см. в КомпьютерПресс №№6-8, 1991). Использование для сжатия файлов утилит-архиваторов, подобных PKZIP, LHA или ARJ, требует обязательного выполнения операции распаковки файла перед его дальнейшим использованием. Кроме того, всегда необходимо иметь достаточный запас свободного дискового пространства для хранения распакованной версии файла. Операция распаковки также отнимает какое-то время (не столько на саму распаковку, сколько на обдумывание и принятие пользователем необходимого решения), не говоря уже о необходимости всегда держать под рукой собственно программу-архиватор.

Главный недостаток невидимых (фоновых) архиваторов заключается в необходимости обязательной загрузки резидентной программы, а в некоторых случаях — и установки специальной платы для работы со сжатыми файлами. Кроме того, большинство фоновых архиваторов в той или иной степени замедляют работу компьютера.

Казалось бы, нет выхода из этого заколдованного круга. Но не все всегда так мрачно, как иногда это представляется на первый взгляд. Оказывается, для .EXE-файлов существует оригинальное и интересное решение по их сжатию, которое лишено практически всех вышеперечисленных недостатков. В чем же оно заключается? Имейте терпение...

Как известно, .EXE-файлы DOS мало заботятся об экономии места на диске. Фирма Microsoft поставляет в составе своих инструментальных продуктов утилиту EXEPACK, уменьшающую размеры .EXE-файлов, оставляя их при этом работоспособными. Достигается это за счет сжатия последовательностей одинаковых символов и оптимизации таблицы настройки адресов (relocation table). Таблица настройки адресов .EXE-файла определяет размещение модулей в памяти при загрузке программы на выполнение.

К сожалению, программа EXEPACK не слишком эффективна в смысле сжатия файлов. В 1982 году фирма Realia (Chicago) выпустила программу SpaceMaker, преобразовывавшую .EXE-файлы в .COM-файлы меньшего размера. Самая первая версия Norton Utilities состояла из .COM-файлов, обработанных SpaceMaker. Поскольку .COM-файл не может быть больше 64 Кбайтов, SpaceMaker нельзя использовать для преобразования больших .EXE-файлов. Хотя программа и выдает предупреждение в случае, если она не может преобразовать какой-либо .EXE-файл, бывает, что полученный после преобразования .COM-файл либо вообще не работает, либо, что хуже, подвешивает машину.

И, наконец, LZEXE

Когда в 1989 году впервые появилась утилита LZEXE, программисты и пользователи персональных компьютеров были просто шокированы ее потрясающими возможностями. LZEXE не только позволяла значительно сократить размер обработанных ею .EXE-файлов. Эта утилита просто "омолаживала" большие и громоздкие программы, заставляя их во многих случаях работать более эффективно.

Подобно EXEPACK и SpaceMaker утилита LZEXE сжимает .EXE-файлы, при этом ее превосходство в эффективности работы и диапазоне применения неоспоримо. Обычно LZEXE сжимает .EXE-файлы на 25-65% от их первоначального объема. (Для сравнения, SpaceMaker сжимает файлы на 2-20%). LZEXE весьма значительно сжимает файлы, ранее упакованные EXEPACK. Однако файлы, еще не "испорченные" EXEPACK, сжимаются лучше, поэтому в пакет программ LZEXE входит утилита UPACKEXE, позволяющая восстановить оригинальный .EXE-файл из файла, упакованного EXEPACK. (Кстати, в процессе обработки файла LZEXE версии 0.91, сообщает, был ли файл ранее упакован с помощью EXEPACK.) По своей эффективности LZEXE сравнима с архиватором PKZIP, с той существенной разницей, что

.EXE-файлы, обработанные ею, остаются полностью работоспособными.

В чем секрет такого чуда? А чуда-то никакого нет. В LZEXE реализована идея сжатия исполняемого файла таким образом, чтобы его распаковка осуществлялась автоматически в процессе загрузки полученного .EXE-файла в ОЗУ компьютера. То есть, после обработки какой-либо программы с помощью LZEXE вы получаете совершенно работоспособный .EXE-файл, но уже значительно меньшего размера. Код распаковщика приписывается в конце сжатого .EXE-файла и занимает всего 330 (!) байтов (для версии 0.91). Работа распаковщика не требует дополнительного пространства ни на диске, ни в ОЗУ. Для этого просто используются те области памяти, которые в дальнейшем будут заняты распакованной программой. Ассемблерный код распаковщика настолько эффективен, что время распаковки файла практически неощутимо.

Здесь складывается одна парадоксальная, на первый взгляд, ситуация. Как выясняется при тестировании, время загрузки многих программ, обработанных LZEXE, меньше времени загрузки тех же программ в оригинальном виде. Чем это объясняется? Дело тут вот в чем. Обычно компьютерные системы компонуются таким образом, чтобы производительность всех их компонентов была приблизительно одинаковой. Нет смысла ставить высокоскоростной жесткий диск со сверхпроизводительным контроллером на компьютер с невысокой тактовой частотой и не слишком мощным процессором. Это значит, что, независимо от класса компьютера, соотношение времени загрузки программы в память ко времени ее распаковки (в случае программы, обработанной LZEXE) будет примерно одинаковым. Здесь-то и обнаруживается, что суммарное время, затрачиваемое на загрузку и распаковку сжатой программы, часто оказывается меньше времени загрузки несжатой программы. Зависит это от размера несжатой программы и степени ее сжатия. Чем больше объем исходной программы и чем выше степень ее сжатия (после обработки LZEXE), тем большим будет выигрыш во времени. А при загрузке программы с флоппи-диска эта разница становится еще более ощутимой.

Программа LZEXE была создана во Франции Фабрисом Белллардом (Fabrice Bellard) и относится к продуктам Public Domain. Первоначально программа имела только французскую документацию, которую впоследствии канадские пользователи перевели на английский. Все сообщения по ходу работы программа также выдает на французском языке. Пользоваться программой очень просто: необходимо в командной строке DOS дать команду

LZEXE имя_файла ,

где параметр "имя_файла" определяет обрабатываемый .EXE-файл. Расширение файла .EXE принимается по умолчанию.

Имейте в виду, что некоторые исполняемые файлы являются .EXE-файлами лишь по своему имени. Операционная система DOS определяет формат исполняемого файла не по расширению файла, а по его заголовку. Так, .EXE-файл должен начинаться символами "MZ", за которыми следует информация о длине файла, размере занимаемой оперативной памяти и т.д. Таким образом, если вы переименуете .COM-файл в файл с расширением .EXE, то DOS этого не заметит, а LZEXE откажется обрабатывать такой файл. Однако пакет LZEXE предлагает способ упаковки и .COM-файлов. Для этого нужно лишь преобразовать .COM-файл в .EXE-формат с помощью утилиты COMTOEXE, выполняющей действия, противоположные EXE2BIN. В дальнейшем полученный .EXE-файл обрабатывают обычным способом.

В процессе обработки .EXE-файла LZEXE переименовывает оригинальный (несжатый) файл в файл с расширением .OLD. Кроме того, утилита создает на диске временный файл с именем LZEXE.TMP, в который записывает код сжатого файла. Лишь после успешного завершения всех операций утилита пере-

именовывает файл LZEXE.TMP в файл с первоначальным именем.

При сжатии файла LZEXE использует алгоритм Лемпела-Зива с кольцевым буфером. Кодировка позиций и длин повторяющихся последовательностей оптимизируется с помощью дополнительного алгоритма, основанного на методе Хаффмана. Утилита может обрабатывать .EXE-файлы в весьма широком диапазоне за счет возможности настройки до 16000 перемещаемых адресов. LZEXE версии 0.91 работает быстрее, чем LZEXE 0.90 и не требует памяти больше, чем это нужно для загрузки оригинальной (несжатой) программы. В версии 0.90 была предусмотрена встроенная защита против вирусного вмешательства, осуществлявшаяся проверкой значения CRC. В версии 0.91 эта опция отсутствует, вследствие чего повысилось быстродействие, и код распаковщика уменьшился с 385 до 330 байтов (проверка CRC теперь выполняется только для кода самого распаковщика).

Очевидно, не все .EXE-файлы могут (и должны) быть сжаты с помощью LZEXE и других подобных программ. Поскольку LZEXE изменяет таблицу настройки адресов, те .EXE-файлы, которые используют подгружаемые оверлейные структуры, не будут работать корректно. При обнаружении оверлейных структур LZEXE выдает предупреждение. Помимо этого, после обработки LZEXE не будут работать программы, проверяющие свой размер или свою целостность. По какой-то причине не работают файлы, скомпонованные редактором связей PLINK (например, файлы FoxBASE или Clipper). Однако файлы, скомпилированные Clipper´ом и скомпонованные редактором связей TLink фирмы Borland программа LZEXE обрабатывает правильно. Существует еще одна проблема. Программы, конфигурация которых сохраняется внутри самого .EXE-файла, могут быть сжаты с помощью LZEXE, однако не могут быть впоследствии переконфигурированы. Поэтому всегда необходимо сохранять их оригинальную версию.

Многие популярные пакеты программ превосходно работают после их сжатия LZEXE. Это, например, dBASE III Plus, Framework II, программы Clipper (скомпонованные TLink), Fastback Plus, Turbo C 2.0, WordPerfect 5.0, 5.1, WordStar 5.0 и так далее. Некоторые другие программы не могут быть сжаты по указанным выше причинам. Сюда относятся Lotus 1-2-3 версии 2.01 (из-за наличия оверлеев), dBASE IV версии 1.1, Harvard Graphics, First Publisher. Автор LZEXE утверждает, что если сжатая .EXE-программа нормально загружается и начинает функционировать, то в дальнейшем она будет надежно работать.

Несмотря на некоторые ограничения в использовании LZEXE, на диске каждого пользователя существует множество программ, которые могут быть с ее помощью эффективно сжаты. Это не только освободит место на вашем жестком диске, но в ряде случаев и увеличит скорость загрузки программ.

А.Синев

По материалам:
Fabrice Bellard, Guide to LZEXE.
P.L.Olympia, “Winning the Disk Bulge Battle“, LanTimes, June 1990.

Оказалось, что новые версии PC Tools и Norton Utilities, предназначенные “специально для DOS 5.0“, были плохо совместимы с этой операционной системой. Сейчас, после получения комментариев пользователей, обе фирмы бесплатно рассылают исправленные версии всем пострадавшим.

Вместо PC Tools 7.0 фирма Central Point Software рассылает всем купившим эту программу ее новую версию 7.1. Обнаружились несовместимость с рядом видеоадаптеров, мышей, а так же проблемы с курсором при работе под MS-DOS 5.0.

У небольшого числа пользователей — 20 или 25 из нескольких тысяч — модуль установки PC Tools вызывал порчу и потерю информации на жестком диске. Проблемой был конфликт между архивирующим модулем PKLite и DOS 5. Сейчас компания PKWare исправила эту ошибку в PKLite версии 1.12.

Исправленная версия Norton Utilities 6.01 посылается пользователям версии 6.0. Эта программа имела похожие проблемы с пятым DOS´ом. Спецификой проблем были конфликты между программами обслуживания кэш-памяти в Norton´е и в DOS´е. У некоторых пользователей это вызывало потерю всех данных на винчестере. Symantec заявляет, что все проблемы такого рода исправлены.

Newsbytes News Network, 23 September 1991

Новая третья версия пакета Clarion Professional Developer будет выпущена в первом квартале 1992 г. Новая версия сможет работать с файлами Btrieve, Ctree, xBase, Netware SQL, и Oracle SQL без преобразования их в формат Clarion. Будет поддерживаться и архитектура клиент-сервер.

Достигнуты также улучшения в работе с экраном, позволяющие отображать смешанные текстовые и графические картинки, добавлена возможность работы с виртуальным экраном, превышающим по размерам дисплей монитора, поддерживаются режимы EGA/VGA с увеличенным количеством линий, мышка, а также элементы оконного интерфейса.

Clarion 3.0 научится работать с расширенной EMS-памятью. Designer, генератор прикладных программ, будет сделан менее интеллигентным, а модельные файлы, используемые Designer´ом — более умными, что даст возможность генерировать более гибкие программы.

Newsbytes News Network, 23 September 1991

Господа, мы ни черта не знаем об Intel! И это действительно так. Большинство из нас с трудом припомнит десяток процессоров этой фирмы, хотя на самом деле только семейств около семнадцати. И это не считая сопроцессоров. Кстати, самый популярный в нашей юной прекрасной стране процессор 8080 (у нас 580ВМ80А) давно снят с производства и используется только для ремонта реликтовых систем. Так что его можно не считать...

Новости от Intel

Итак, фирма Intel изготавливает несколько типов процессоров. Это процессоры для микрокомпьютеров, "суперкомпьютерные" процессоры — скоростные RISC-процессоры для мощных систем, и, наконец, море процессоров для создания микроконтроллеров, встраиваемых в самые различные устройства. Кроме того, Intel делает множество периферийных устройств, память, программные и аппаратные средства для отладки микрокомпьютерных систем и самые разные платы: сетевые, телекоммуникационные (например, знаменитая плата SatisFAXtion), платы расширения памяти и т.д.

Мы расскажем о некоторых процессорах Intel, завоевавших большую популярность во всем мире и теперь доступных в Советском Союзе. Причем, на последней пресс-конференции представитель фирмы заявил, что их можно использовать даже в военных изделиях. Необходимое для этого разрешение КОКОМ, вполне может быть получено.

Теперь, обратившись в фирму PC Center Techno, которая является дистрибьютором Intel в нашей стране, можно получить не только процессоры, но и подробнейшую информацию как о продуктах Intel, так и о периферийных устройствах, изготавливаемых другими фирмами. Кроме того предоставляются все услуги, оказываемые потребителям в остальных странах мира. Уже заключены договоры на поставку микросхем Intel на сумму свыше 100000 долларов.

До недавнего времени изделия Intel везли в нашу страну через восточные страны. Прямые поставки в СССР позволят снизить цену процессоров и, соответственно, систем на их базе. Кроме того будут соблюдаться сроки поставки, а те дополнительные услуги, о которых мы говорили выше, станут сильным стимулом для разработки отечественных систем на современнейшей элементной базе.

Итак, что же нового предлагает Intel потребителям?

Новые процессоры фирмы Intel для компьютеров

Произошла смена лидера в семействе микропроцессоров Intel для компьютеров. В конце июня фирма официально представила новый чип по имени 486DX/50, работающий значительно быстрее своего предшественника 80486/33. Intel считает, что новый процессор фактически обеспечивает производительность компьютеров класса больших ЭВМ, работая в два раза быстрее процессора 80486 при тактовой частоте 33 МГц.

Такие характеристики достигнуты благодаря удачной схеме кристалла процессора, в которой наглядно реализован девиз "Меньше — значит быстрее". Процессор 486DX/50 по размерам существенно меньше, чем 80486/33; он выполнен в виде трехслойной микросхемы, тогда как 80486/33 реализован как двухслойная конструкция. В результате в новом 486DX/50 электрические сигналы проходят существенно меньшие расстояния и операции выполняются быстрее. Кроме того, с новым процессором поставляется дополнительная кэш-память объемом 256 Кбайт.

Многим ли понадобится мощь этого монстра? Вероятно, нет, хотя можно вспомнить, что после появления 80486 ряд экспертов предсказывали, что этот процессор

не найдет своего рынка из-за сверхбольшой, можно сказать, никому не нужной, производительности. Жизнь рассудила иначе, как, скорее всего, получится и на этот раз. Ряд компаний уже объявил о разработке систем на основе нового процессора, а первой из них стала фирма Compaq. Она уже выпустила компьютер Compaq Descpro 486/50L, снабдив его несколько невнятным комментарием, что появление этого компьютера поможет компании снизить цены на другие системы.

О разработке систем на базе нового процессора уже объявили Dell Computer и Tandon, ожидается, что до конца года об этом объявят Everex и ALR.

В сентябре, во время подготовки этого номера КомпьютерПресс, новый процессор продавался фирмой еще в ограниченных количествах и стоил примерно на 200 долларов дороже, чем 486/33, из чего можно сделать вывод, что первые компьютеры на базе этого кристалла, будут стоить где-то от 8000 до 15000 долларов.

Сегодня микропроцессор Intel 80386SX утвердился на рынке как процессор для ноутбуков. Однако в лице нового изделия неутомимой фирмы Intel он приобрел достойного брата-конкурента. Полное имя этого брата: Intel´s 386SL Microprocessor Superset, в семье же его кличут просто 386SL. Если полное имя перевести на русский язык, то получится что-то типа "микропроцессорный супернабор 386SL фирмы Intel". И действительно, этот кристалл, по сути дела, не просто процессор — это, скорее, почти законченный компьютер — процессор плюс периферия. В сентябре фирма Zenith Data System начала поставки компьютера MastersPort 386SL — первого ноутбука, использующего этот микропроцессор. 386SL был спроектирован именно для использования в портативных машинах, поэтому в его схеме учтены некоторые весьма специфические вещи. В частности, в его состав входит экономичная схема ввода-вывода 82360SL, в результате обеспечивается производительность процессора 80386/20, а жизнь аккумуляторов продлевается в два раза. Кроме того, он почти на 40 процентов меньше своего старшего брата 80386SX, благодаря чему на плате остается дополнительное место. Но главным его преимуществом является возможность управления питанием, в результате чего производители, проектирующие ноутбуки, могут теперь использовать образующиеся резервы для питания других элементов компьютера. Например, фирма Zenith Data System утверждает, что время работы аккумуляторов компьютера MastersPort 386SL продлено до 8 часов. Эта машина стоящая 4999 долл. обладает рядом возможностей управления питанием, например, в ней есть режим "отдыха", в котором компьютер при необходимости может находиться в течение двух недель, не прерывая работы прикладных программ.

Однако, надо учесть, что многим покупателям срок службы батарей не столь важен, как, скажем, производительность. Такие покупатели обратят свое внимание на процессоры фирмы AMD AM386SX и AM386SXL (микросхемы с малой потребляемой энергией, совместимые с процессором 80386SX фирмы Intel), работающие на частоте 25 МГц. В связи с этим, многие эксперты считают, что в 1992 г. компьютеры, построенные на базе процессора 386SL, завоюют не более 15 процентов рынка ноутбуков.

Тем не менее, появились сообщения, что фирма Dell приступила к работам над созданием ноутбука с цветным экраном на базе процессора 386SL, а на выставке Comdex в октябре этого года будут представлены компьютеры фирм Toshiba, AST и NEC, построенные на базе этого процессора.

Новинки Intel для микроконтроллеров

30 сентября 1991 года фирма Intel объявила о начале продаж трех новых процессоров семейства 80C186.

Процессор 80186 не завоевал широкой популярности на рынке персональных компьютеров (впрочем, он и не был ориентирован на такое применение), так как выпущенный в следующем 1983 году процессор 80286 был, в отличие от 80186, существенным шагом вперед по сравнению с предыдущими процессорами фирмы Intel. Однако, этот процессор очень популярен у разработчиков встроенных контроллеров, предназначенных для управления работой самых разных устройств — начиная от стиральных машин и модемов и кончая системами современных самолетов. Популярность серии особенно возросла, когда в 1987 году появились процессоры 80C186, потреблявшие значительно меньше энергии, нежели их предшественники. Было даже несколько вполне успешных разработок портативных компьютеров на базе процессоров этого семейства, довольно широко они применяются в электронных органайзерах (планировщик рабочего времени плюс "умная" записная книжка плюс калькулятор плюс многофункциональные часы плюс многое другое). В 1991 году во всем мире будет использовано около 10 миллионов процессоров семейства 186, произведенных на заводе Intel в городе Чандлер.

В новых процессорах этого семейства реализовано несколько дополнительных функций, расширяющих возможные области их использования. Строго говоря, процессоров не три, а восемь — каждый из трех доступен как в 16-разрядном варианте (с шестеркой на конце), так и в 8-разрядном (с восьмеркой), а 80C186EA/188EA доступен еще в варианте с 3-х вольтовым питанием под названием 80L186EA/188EA.

В целом семейство 80186 — это 16 различных процессоров.

Процессор 80C186EC/80C188EC

Это, в сущности не просто микропроцессор, а мощная однокристальная микроЭВМ. В одной мик-

росхеме вместе с процессором собрана практически все основная периферия, используемая для работы с 80186. В итоге разработчик может отказаться от использования 20 дополнительных корпусов, оставив на плате только 80С186ЕС и память. При этом снижаются размер устройства, его масса, потребляемая энергия и стоимость, растет надежность.

Кроме собственно стандартного процессора семейства С186 кристалл содержит встроенные тактовый генератор, три таймера, четыре канала прямого доступа к памяти, контроллер прерываний, два последовательных порта, еще 22 программируемых порта, которые могут быть использованы для соединения с клавиатурой, светодиодами, модемами, принтерами и прочими устройствами. Этот процессор имеет очень ценную функцию: использование режимов сохранения энергии, что особенно важно при создании систем с автономным питанием.

Процессор выпускается в вариантах для работы с тактовой частотой либо 13, либо 16 МГц. Адресное пространство памяти составляет 1 Мбайт, адресное пространство внешних устройств — 64 Кбайта. Процессор позволяет использовать чередование памяти, извлекая из нее данные либо побайтно, либо словами. К нему можно без проблем подключить сопроцессор 80С187.

В нормальном режиме процессор потребляет 82 мА при тактовой частоте 13 МГц. Кстати, первый процессор 80186 потреблял 320 мА при частоте 8 МГц.

Режимы экономии энергии позволяют существенно продлить время работы в автономном режиме. В данном процессоре предусмотрено три режима экономии энергии: режим "отдыха", режим сохранения энергии и режим отключения энергии.

Режим отдыха — это перевод центрального процессорного блока в "замороженное" состояние (фактически, на процессор не подается тактовый сигнал). При этом вся периферия, расположенная на кристалле, продолжает работать. Экономия энергии составляет от 30 до 50%. Потребляемый ток снижается до 58 мА. Процессор "спит" до тех пор, пока не появится либо немаскируемое прерывание, либо сигнал RESET, либо сигнал NMI. Такой режим полезен для систем, большую часть времени пребывающих в состоянии ожидания, например для различных коммуникационных устройств.

Режим сохранения энергии использует тот, факт, что при уменьшении тактовой частоты снижается потребляемая энергия. При снижении частоты вдвое потребляемая мощность падает на 40-45%. Реализуется данный режим делением тактовой частоты на 1, 4, 8, 16, 32 или 64. При таком переключении задача, разумеется, продолжает выполняться и никаких сбоев происходить не должно.

Третий режим — режим отключения энергии основан на отключении всех тактовых генераторов. При этом "замирает" как собственно процессор, так и вся периферия. Потребляемый ток очень мал — не более 100 мкА. "Пробуждение" наступает при появлении либо сигнала NMI, либо сигнала RESET. Но есть третий, более интересный путь: вы можете задать время пребывания процессора в отключенном состоянии с помощью внешнего конденсатора. Внутреннее состояние сохраняется неизменным и возвращается в момент "пробуждения".

В этом процессоре есть одно очень интересное устройство — "сторожевой" таймер (Watchdog Timer). По замыслу, он предназначен просто для контроля работоспособности процессора. Идея заключается в том, что таймер имеет регистр, содержимое которого уменьшается на единицу каждый такт синхронизации. При обнулении этого регистра на отдельный выход в течении четырех тактов выдается логический нуль. Регистр 32-разрядный, его можно устанавливать всего один раз, после чего таймер будет работать с этим значением в цикле до тех пор, пока не выключится питание.

Исходя из описания работы этого устройства напрашиваются две мысли. Во-первых, при желании, его можно использовать в качестве четвертого таймера. Во-вторых, с его помощью можно "будить" процессор, находящийся в одном из состояний сохранения энергии, через определенные промежутки времени. Эти промежутки могут быть весьма большими, если вспомнить, что регистр этого таймера 32-разрядный.

Еще одно важное периферийное устройство 80С186ЕС — встроенный блок регенерации ОЗУ с расширенными возможностями.

Все характеристики этого процессора нацелены на то, чтобы на его базе можно было построить компактные высоконадежные устройства, обладающие высокой производительностью при приемлемой стоимости. Его использование поможет повысить качество ваших контроллеров, используемых в принтерах, адаптерах локальных сетей, мощных модемах, электронных АТС и других коммуникационных устройствах, в контрольно-измерительных приборах, в самых разнообразных устройствах с автономным питанием.

Процессор 80С186XL/188XL

Этот процессор изготовлен по новой 1-микронной CHMOS-технологии, запатентованной фирмы Intel. Это самый производительный процессор семейства 80186; его тактовая частота — 20 МГц. При его разработке ставилась цель добиться полной совместимости с оригинальным процессором 80С186. Поэтому 80С186XL, обладая лучшими характеристиками, полностью совпадает с 80С186 по расположению выводов, набору функциональных узлов и т.д. Фактически, это "освеженный" 80С186, с увеличенной на четверть производительностью и сниженной на 50% потребляемой мощностью.

Процессор выпускается в 10, 12.5, 16 и 20 мегагерцовых вариантах.

Этот кристалл содержит встроенные тактовый генератор, три таймера, два канала прямого доступа к памяти, контроллер прерываний, встроенный блок регенерации ОЗУ. Возможно подключение сопроцессора 80C187. Адресное пространство памяти — 1 Мбайт, внешних устройств — 64 Кбайта.

Процессор 80C186XL может использовать простейший режим сохранения энергии — деление тактовой частоты на 1, 4, 8 или 16. Потребляемый ток 21 мА при тактовой частоте 8 МГц и 52 мА при 20 МГц.

Это вторая по счету переработка 80186. Этот процессор был выпущен в 1982 году, затем в 1987 появился его CMOS-вариант 80C186, и вот теперь, 30 сентября 1991 объявлено о рождении 80C186XL. Данный процессор стал базовым для дальнейшего развития семейства 186.

80C186XL с его высокой производительностью особенно подходит для систем, связанных с обработкой данных. На его базе можно строить факсы среднего и высокого класса, скоростные модемы, контроллеры винчестеров, и многие другие устройства.

Процессоры 80C186EA/188EA и 80L186EA/188EA

В целом этот процессор аналогичен 80C186XL, но дополнительно содержит режим "отдыха" и режим отключения энергии. Процессор 80C186EA/80C188EA выпускается в вариантах для тактовых частот 12.5, 16 и 20 МГц. Потребляемый ток 21 мА при тактовой частоте 8 МГц и 52 мА при 20 МГц. В режиме отдыха он снижается до 16 или 39 мА соответственно. В режиме отключения энергии потребляемый ток не превышает 50 мкА.

Существует низковольтный вариант этого процессора 80L186EA/80L188EA. Он работает при напряжении питания от 2.7 В до 5.5 В. Тактовая частота для этого процессора — 8 МГц. Потребляемый ток составляет 38 мА при номинальном напряжении питания, равном 3 В; в режиме отдыха он снижается до 20 мА. В режиме отключения энергии, как и у пятивольтового варианта, не превышает 50 мА.

Этот процессор ориентирован на использование в устройствах с батарейным питанием. (Представьте себе нечто с мощным интеллектом, работающее от двух батареек А316...) Этот процессор с успехом можно использовать в портативных приборах. Например, в карманном сетевом адаптере для ноутбука или в крутом факс/модеме, который можно сунуть в другой карман.

Напоследок расскажем еще об одном представителе этого семейства, выпущенном в середине прошлого года.

Процессоры 80C186EB/188EB и 80L186EB/188EB

Это более мощный процессор, также ориентированный на применение в аппаратуре с батарейным питанием. Кроме стандартных элементов, 80C186EB содержит два последовательных порта, узел регенерации динамического ОЗУ с расширенными функциональными возможностями, 16 программируемых портов, режимы отдыха и отключения энергии.

Пятивольтовый вариант работает с тактовыми частотами 8, 13 и 16 МГц. Трехвольтовый — с частотой 8 МГц.

Процессор 80L186EB/188EB — самый экономичный из всего семейства. В активном режиме он потребляет всего 14 мА, а в режиме отдыха 10мА. При отключении энергии — как и его братья, не более 50 мкА.

Как с этим живут

Несколько слов о том, как работают с процессорами Intel в нормальных странах. Кроме того, что в нормальных странах есть нормальные приборы, с которыми можно нормально работать, фирма выпускает специальные средства для отладки изделий на базе своих процессоров. В том числе на безе семейства 80186.

Итак, вы можете пользоваться трансляторами не только с макроассемблера, но и с таких языков высокого уровня, как Pascal, C, Fortran, PL/M. Причем C и Fortran полностью соответствуют стандартам ANSI, а Pascal — стандарту ISO. К каждому транслятору придается библиотека функций, оптимизированная для данного процессора и имеющая дополнительные функции для обеспечения максимально гибкого управления им. Кроме того, редактор связей умеет генерировать код специально для ПЗУ, а библиотеки могут быть зашиты в ПЗУ.

Каждый транслятор генерирует всю информацию, необходимую для работы с символическим отладчиком и внутрисхемным эмулятором. Кстати, эмулятор тоже есть, причем для каждого процессора. Это правило фирмы Intel — важно не только сделать микросхему, важно помочь разработчикам использовать все ее возможности с минимальной головной болью.

В отладочном комплекте поставляется плата эмулятора и отладчик Paradigm DEBUG/RT, представляющий собой снабженный специализированными инструментами Turbo Debugger.

Обеспечивается полная информационная поддержка потребителей, включающая не только предоставление справочных материалов, но и консультации.

Современные процессоры фирмы Intel, пожалуй, не только обладают высокой степенью интеграции, но и, что очень важно, они хорошо интегрированы. Думается, наша статья рассказала об этом.

И.Вязаничев, Б.Молчанов

По материалам, любезно предоставленным фирмой Intel.

Специальный выпуск, посвященный электронной почте и средствам, связанным с ее эксплуатацией, вызвал большой отклик среди наших читателей. Поэтому сегодня КомпьютерПресс начинает публикацию цикла статей, посвященных наиболее современным средствам телекоммуникации — глобальным сетям. Мы расскажем о том, как работает электронная почта на примере наиболее популярной в нашей стране сети RELCOM. Кроме того в дальнейшем постараемся рассказать и о других сетях, функционирующих в нашей стране.

Сеть RELCOM и электронная почта

Наконец-то наша Родина сделала решительный скачок в XX век, вдогонку развитым капиталистическим странам. На этот раз в сфере электронных коммуникаций. Чуть больше года назад советская сеть электронной почты RELCOM установила связь с сетью EUnet Европейской Ассоциации Открытых Систем EurOpen и советские пользователи стали полноправными членами международного сетевого сообщества. Теперь сообщение, отправленное пользователем сети RELCOM, доставляется адресату практически в любой точке Земного шара в среднем за полтора часа.

Как устроена сеть

RELCOM — это сеть передачи сообщений, объединяющая почтовые компьютеры в основном на территории Советского Союза (теперь уже бывшего — так что сеть неожиданно стала интернациональной). На правах национальной сети RELCOM является частью европейской сети EUnet, поэтому на абонентов сети RELCOM распространяются соглашения об обмене почтовыми сообщениями, существующие между сетью EUnet и другими глобальными сетями (Internet, UUnet, BITNET, CompuServe, MCImail и др.) Подключившись к RELCOM, пользователь получает возможность переписываться по электронной почте как с другими абонентами RELCOM, так и с абонентами международных сетей.

Сеть состоит из нескольких основных компонентов. Конечный пользователь имеет компьютер и специальное программное обеспечение, позволяющее ему через модем передавать сообщения на узловую машину. Это первый элемент системы.

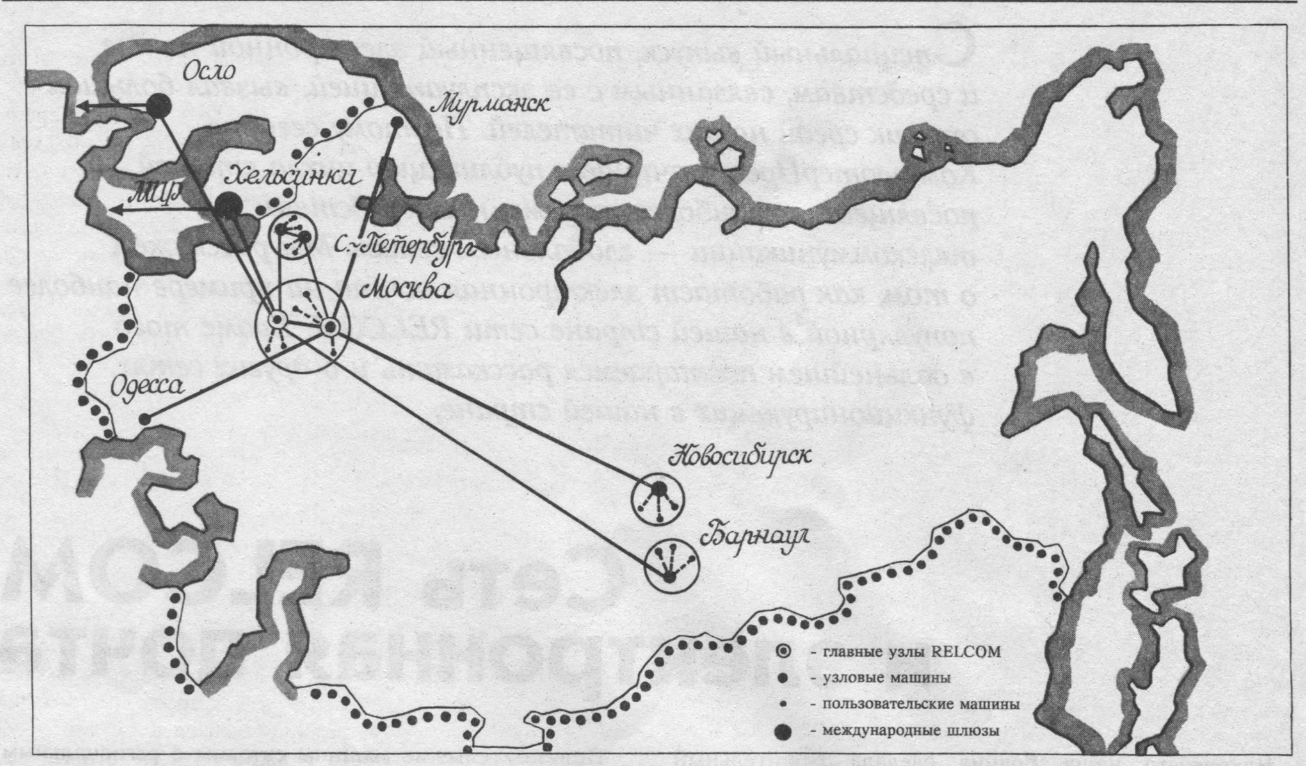
Пользовательские машины связаны с региональным узлом при помощи модемов по обычным телефонным линиям. Пользователь со своего компьютера может в любое время позвонить на узловую машину, получить адресованные ему сообщения, хранящиеся на ней, и отправить свои.

Узловые машины, расположенные в крупных городах, обеспечивают обмен письмами и распространение сообщений телеконференции в своих телефонных регионах. Региональные центры RELCOM соединяются выделенными каналами связи или используют каналы специализированной телефонной сети, что, в комплексе с использованием высокоскоростных модемов, обеспечивает возможность быстрой передачи больших объемов информации между узлами. В качестве узловых обычно используются мощные миниЭВМ, работающие под управлением операционной системы класса UNIX. Узловые компьютеры работают круглосуточно, в итоге пользователь оказывается свободен в выборе времени работы с узлом.

RELCOM образовалась как небольшая сеть, использовавшаяся разработчиками и пользователями операционной системы ДЕМОС, но, быстро развиваясь, она вобрала в себя сотни машин, работающих под управлением различных систем семейств UNIX, MS-DOS и других. Сейчас в сети работает несколько тысяч пользователей из более чем 500 организаций, расположенных в 80 городах Союза.

Что можно делать в RELCOM

Абонент сети RELCOM может пользоваться электронной почтой для обмена сообщениями не только с абонентами RELCOM, но и с людьми, работающими



на других сетях. Кроме того возможно получать сообщения телеконференции USENET (об этом ниже) по интересующим его темам и отправлять свои собственные сообщения в телеконференцию, есть доступ к публичным архивам, существующим на больших машинах как в RELCOM, так и в других связанных с ней сетях во всем мире.

Ну, а теперь расскажем по порядку о каждом из элементов сети.

Электронная почта

Электронная почта по своему устройству во многом напоминает обычную, "бумажную" почту. Пользователи обмениваются сообщениями, очень похожими на письма. Текст письма вводится с клавиатуры или "приклеивается" из файла, снабжается заголовками, содержащими адрес получателя, дату отправления, тему, обратный адрес — все то, что пишется на почтовом конверте, — и отправляется адресату. Сетевой адрес тоже имеет много общего с обычным почтовым адресом, только вместо имени человека, страны, города, улицы и номера дома содержит стандартный код страны, код города, название организации и имя машины, на которой пользователь зарегистрирован, а также регистрационное имя пользователя на этой машине. Такой принцип адресации используется в самой большой сети мира — в Internet. Он стал своего рода стандартом de-facto, его придерживаются очень многие сети, в том числе и RELCOM. Например, адрес mike@kremlin.msk.su обозначает, что письмо адресовано пользователю mike в организации Kremlin, Москва, СССР (за достоверность адреса не ручаюсь).

Письмо, отправленное с локальной машины, поступает на почтовую машину регионального сетевого узла, которая по адресу определяет маршрут, по которому нужно доставить письмо. Маршрут зависит не только от местоположения конечного адресата, но и от загруженности линий и конфигурации сети в данный момент, поэтому указывая адрес, вы определяете только то, куда письмо должно дойти, но не оговариваете, каким путем. Это важный момент. Он означает, что вам не нужно знать множество подробностей, связанных с циркуляцией сообщений в сети. Система сама будет искать оптимальный (на данный момент!) путь к адресату, даже если для этого вашему письму придется пробраться сквозь десяток промежуточных сетей.

Письмо передается на машину адресата и "падает" в его почтовый ящик — то есть записывается в файл, который система управления электронной почтой считает почтовым ящиком этого пользователя. При входе пользователя в систему, она сообщает, что ему пришла почта. Пользователь может "забрать" и прочитать ее, вызвав программу просмотра почты. (Это похоже на ту процедуру, которую вы выполняете каждое утро, вынимая письма из почтового ящика). Эта программа позволяет прочесть полученные сообщения, рассортировать их, ответить на них, уничтожить некоторые из них или записать в архив — как бы разложить по ящикам стола (по папкам), в общем, сделать практически все то, что мы обычно делаем с обычными письмами. Электронным письмом нельзя, однако, раскуривать трубку.

При составлении письма можно включать в него отрывки других сообщений, текстовые файлы, нетек-

стовую информацию (графические файлы, объектные коды и т.п.) в закодированном виде. Можно очень просто разослать одно и то же письмо нескольким адресатам по списку или отправив одному адресату письмо, разослать другим его копии. Тут мы подошли к еще одной интересной возможности электронной почты — к списку рассылки.

Если несколько человек намереваются затеять дискуссию по электронной почте, или если кто-то собирается рассылать информацию постоянному кругу своих "слушателей", создается так называемый список рассылки. При этом организатор этого дела составляет список адресов всех участников дискуссии, присваивает ему имя, после чего все сообщения, отправляемые на это имя, распространяются по списку.

Например, во время недавнего правительственного кризиса, когда возникла необходимость наладить обмен сообщениями между Российским правительством и местными органами власти в разных городах, был организован список рассылки, названный relcom.politics, в него включили всех активных пользователей сети и отправляли по этому списку правительственные сообщения, сводки информационных агентств и просто информацию о происходящем, а уж каждый, получавший такую информацию, распространял ее как мог в своем регионе. Кстати, почтовые списки весьма успешно используются и в мирное время.

Телеконференция

Если электронная почта напоминает "бумажную" почту, то телеконференция больше всего похожа на всемирную доску объявлений или на газету, публикующую исключительно письма читателей. В отличие от электронной почты автор посылает сообщение не конкретному адресату, а всем желающим его прочесть. Сообщение рассылается на все соседние (в сети) машины, они, в свою очередь, передают его своим соседям, в итоге сообщение очень быстро расползается по всему миру. Пользователи на машине, через которую проходят сообщения телеконференции, при желании могут его прочесть. Чтобы читателю легче было определить, хочет ли он читать это сообщение, кроме темы указывается еще и группа (круг интересов), к которой это сообщение относится. Поскольку каждый день в телеконференции проходит огромное число сообщений, читатели заранее указывают, какие группы их интересуют. После этого им остается только выбрать нужные сообщения уже внутри этих групп по строчке темы. Действительно, прочесть и понять все приходящие ежедневно 6-7 мегабайт информации, идущей во всех двух с лишним тысячах телеконференций вряд ли под силу человеку (ведь это где-то 3000 машинописных страниц!).

Большинство машин, работающих в RELCOM, не имеет возможности получать все сообщения телеконференции, так как это требует от компьютера наличия очень больших ресурсов. Фактически всю телеконференцию получают только несколько крупных региональных узлов, а для пользователей на небольших машинах предусмотрена возможность выбирать сообщения и получать их по электронной почте через специализированные "почтовые серверы телеконференций".

Почтовый сервер телеконференций, с точки зрения пользователя, — это адресат сети, некий клерк, который принимает письма-заказы и обрабатывает их. Его можно попросить прислать список полученных сообщений телеконференции, прислать выбранные сообщения в виде обычных электронных писем или переправить сообщение абонента в телеконференцию. Сейчас такие серверы есть на большинстве региональных узлов. Их адреса можно узнать у администратора соответствующего узла. Кроме того, есть два сервера, получающих все телеконференции и обслуживающих остальные машины сети. Они проживают по адресам news@hq.demos.su и news@kiae.su. Естественно, на их долю выпадает очень много работы, человеку она не под силу, поэтому такие серверы выполняют ее автоматически.

Архивы

Такие же автоматы обслуживают публичные архивы файлов, расположенные на наиболее мощных узловых машинах и доступные пользователям через электронную почту. Архивный сервер по запросу пользователя высылает список файлов, хранящихся в архиве, пересылает отдельные файлы и выполняет некоторые другие операции. Обычно архивы содержат тексты некоммерческих программ, документацию, современный фольклор и т.п. Адреса архивных серверов периодически появляются в телеконференциях.

Зачем все это нужно?

По данным опроса, проведенного недавно в сети, подключение к RELCOM не облегчает пользователю жизнь. Большинство опрошенных сознались, что за время их работы в сети (обычно 6-12 месяцев) круг их общения сильно расширился, стали интенсивнее личные и деловые контакты, на людей обрушился небывалый поток информации. Как средство от информационного голода сеть не сравнится ни с чем, поскольку абонент сети может в любой момент получить гораздо больше информации, чем он сможет прочитать, а тем более осмыслить. Это сразу создает проблему выбора действительно нужной информации из колоссального объема получаемой.

У пользователей сети неожиданно возникли проблемы чисто психологического характера. Дело в том, что сеть — это главным образом не провода и компьютеры, а люди, которые хотят и могут друг с другом общаться. В сетевом сообществе свои правила поведения (отличные от обычных, ведь вы не видите человека, с которым общаетесь, часто даже не знаете, кто он!), свои неформальные группировки. Нужно некоторое время повариться в этом котле, чтобы чувствовать себя здесь уверенно. Впрочем, никто из опрошенных не сожалеет о том, что по своей воле попал в этот новый необычный мир.

П.Антонова



"Многие вещи нам непонятны не потому, что наши понятия слабы, но потому, что сии вещи не входят в круг наших понятий".

К.Прутков

Заглянем на диск

Каждый, кто впервые попадает в удивительный мир персональных компьютеров, непременно сталкивается буквально с лавиной новых английских и русских терминов. Часть из них пока "неузаконена" (по крайней мере, у нас), поэтому под разными названиями нередко скрыты одни и те же понятия. Это зачастую приводит к путанице даже среди вполне квалифицированных прикладных программистов. Одно из наиболее типичных терминологических заблуждений, с которым приходится сталкиваться на практике, связано с первым физическим сектором на винчестере и флоппи-диске (сектор 1, головка 0, цилиндр 0).

"Если на клетке слона прочтешь надпись "буйвол", не верь глазам своим".

Опуская подробности работы POST BIOS (Power-on Self Test) IBM-совместимого компьютера, обратимся к другой важнейшей процедуре BIOS — подготовке начальной загрузки модулей DOS. Эту функцию осуществляет ROM Bootstrap Routine — программа начальной загрузки, хранящаяся в ПЗУ, т.е. BIOS. Иногда эту программу называют просто — "Начальный Загрузчик". Вышеназванная программа первым делом пытается считать сектор 1 (головка 0, цилиндр 0) с устройства A: в оперативную память компьютера по адресу 0:7C00h. В случае, если это устройство не готово, делается попытка прочитать тот же физический сектор, но уже с первого винчестера (устройство C:). Первый физический сектор на дискете имеет несколько названий: Boot Sector, Boot Record, Корневая Запись и, опять же, Начальный Загрузчик. С названиями такого же сектора на винчестере тоже не соскучишься: Master Boot Record, Partition Table Sector, Boot Sector, Master Boot Sector, Блок Начальной Загрузки. Условимся пока (хотя это дело вкуса) называть эти физические сектора так: для дискеты — просто Boot-сектор, а для винчестера — Master Boot Sector.

"Смотри в корень!"

Первым байтом Boot-сектора диска должен быть либо код безусловного перехода JMP (E9h) с последующим 16-битным смещением, либо код "короткого" (short) перехода JMP (EBh) с 8-битным смещением, причем третьим байтом в этом случае является код операции NOP (90h). Заканчивается сектор определенной кодовой комбинацией — сигнатурой — 0AA55h.

Сразу за инструкцией JMP в этом секторе следует 8-байтное поле, резервируемое для идентификации имени и версии OEM (Original Equipment Manufacturer), например, MS DOS 3.3 или PC Tools.

Третьим — главным компонентом Boot-сектора — является BIOS Parameter Block (BPB — блок параметров BIOS). Это важнейшая структура данных, содержащая, в частности, тип носителя (media descriptor), а также другие параметры, характеризующие формат диска (рис. 1).

Последний элемент Boot-сектора диска — это программа, называемая обычно Bootstrap, но, чтобы не путать ее с ROM Bootstrap Routine (и с легкой руки И. Карасика), удобнее назвать ее IPL2 (Initial Program Loading 2). Начальная инструкция JMP в Boot-секторе выполняет переход на точку входа именно этой программы. IPL2 в свою очередь, используя информацию из BPB, определяет, являются ли два первых файла в корневом оглавлении диска модулями DOS. Затем про-

грамма загружает эти файлы в младшие адреса памяти (70:0000h) и передает управление на IO.SYS

Смещение hex	Размер, байт	Содержание	
00h	3	Код инструкции перехода на программу IPL2 E9xxxx или EBxx90	
03h	8	Имя и версия OEM	
0Bh	2	Количество байт на сектор	
0Dh	1	Количество секторов на кластер	
0Eh	2	Количество резервных секторов, включая Boot-сектор	BPB
10h	1	Число таблиц FAT	
11h	2	Максимальное число элементов в корневом оглавлении	DOS 2.0
13h	2	Общее количество секторов на логическом диске	
15h	1	Тип носителя (media descriptor)	
16h	2	Количество секторов в одной FAT	
18h	2	Количество секторов на трек	
1Ah	2	Количество головок	DOS 3.0
1Ch	4	Количество "скрытых" секторов	---
20h	4	Общее количество секторов на логическом диске	DOS 4.0
24h	1	Физический номер диска	
25h	1	Зарезервировано	+ DOS 4.0
26h	1	Сигнатура 29h	
27h	4	Двоичный номер диска	
2Bh	11	Метка диска	
36h	8	Зарезервировано	
3Eh		ПРОГРАММА IPL1	
1FEh	2	Сигнатура 0AA55h	

Рис.1. Формат Boot-сектора.

(IBMBIO.COM). Далее следует процесс инициализации, выполняемый средствами самой DOS.

"Не в совокупности ищи единства, но более — в единообразии разделения".

Если в память компьютера считан с винчестера Master Boot-сектор, то управление передается программе IPL1, которая расположена в его начале (рис. 2). Эта программа сканирует содержание Partition Table (таблицы деления диска), состоящее из четырех 16-байтных элементов, разбитых на поля. Поля содержат информацию о номерах начального и конечного секторов, номерах головок и цилиндров для соответствующего раздела, а также числе секторов, предшествующих разделу и включенных в раздел (рис. 3). К разочарованию пользователей DOS 3.30 надо отметить, что из четырех разделов диска только два могут принадлежать DOS — Primary и Extended, два оставшихся фирма Microsoft благородно резервирует для альтернативных операционных систем (ОС), например, CP/M-86. Пользователи DOS версий от 2.0 до 3.2 могут позволить себе вообще только один раздел DOS на поделенном диске.

Байт поля "Признак загрузки" используется программой IPL1 для выяснения, какой из разделов диска содержит загружаемую ОС. Активный (загружаемый) раздел в этом поле содержит код 80h, остальные разделы должны быть помечены кодом 00h, даже если ОС и могла бы быть загружена из этих разделов. Программа IPL1 считывает сектор, номер которого находится в поле "Начало раздела", а именно, в трех байтах, следующих за кодом 80h. В этих байтах находятся номера головки, сектора и цилиндра стартового сектора раздела. Несколько странное, на первый взгляд, распределение битов в байтах номеров сектора и цилиндра

Байты	Размер байт	Содержание
000 – 1BDh	446	Резервируется для IPL1
1BE – 1CDh	16	Элемент 1-го раздела
1CE – 1DDh	16	Элемент 2-го раздела
1DE – 1EDh	16	Элемент 3-го раздела
1EE – 1FDh	16	Элемент 4-го раздела
1FE – 1FFh	2	Сигнатура 0AA55h

Рис.2. Таблица деления диска (Disk Partition Table).

соответствует формату загрузки регистров перед обращением к функции чтения физического сектора диска прерывания 13h BIOS, обязательно используемого в программе IPL1. Хотя дизассемблирование — дело не для всех увлекательное, без особых усилий в шестнадцатиричном дампе IPL1 можно отыскать, по крайней мере, два обращения к int 13h (байты CD13h). Выбранный таким образом сектор является Boot-сектором активного раздела винчестера, а его содержание аналогично содержанию Boot-сектора флоппи-диска.

"Щелкни кобылу в нос — она махнет хвостом".

Программа IPL1 может выдавать на экран три сообщения. Если Partition Table содержит более одного загружаемого раздела — выдается сообщение Invalid Partition Table; если Boot-сектор активного раздела не удается считать в память — выдается сообщение Error loading operation system; если в Boot-секторе отсутствует сигнатура 0AA55h — выдается сообщение Missing operating system.

"Не будь портных — скажи: как различил бы ты служебные ведомства?"

Теперь еще об одном важном поле элементов Partition Table — "Тип раздела". Код в этом поле указывает, какой именно ОС принадлежит данный раздел. Начиная с DOS 3.0, максимальный размер раздела физического диска может быть увеличен с 16 Мбайтов до 32 Мбайтов, благодаря введению 16-битных элементов FAT (File Allocation Table — таблица размещения файлов на диске). Хотя теоретически DOS 3.0 могла бы управлять диском размером вплоть до 134 Мбайтов. Действительно, если учесть, что максимальное количество кластеров 65535, а каждый кластер содержит 4 сектора размером по 512 байт: 65535*4*512 = 134215680 байт. Но граница в 32 Мбайта диктуется здесь известной структурой BPB в Boot-секторе раздела диска, в которой для общего количества принадлежащих диску секторов отведено лишь два байта (максим. 65535). Таким образом, максимальный размер действительно составляет: 65535*512 = 33553920 байтов.

Для того, чтобы сохранить совместимость с DOS 2.X, в DOS 3.0 была оставлена возможность управлять разделами диска с 12-битными элементами FAT. Поэтому все DOS 3.X-разделы диска размером меньше 16.7 Мбайт используют 12-битные элементы FAT. Так что при использовании разделов диска меньше указанного размера код в поле "Тип раздела" будет соответствовать DOS 2.X. Размеры разделов винчестера, начиная с DOS 4.0, перешагнули границу в 32 Мбайта и теоретически могут достигать фантастической пока цифры — 2048 Мбайтов. Для этого формат BPB был расширен, и под поле общего числа секторов диска было отведено уже 4 байта. Ревностным приверженцам DOS версии 3.30 не следует забывать, что Extended-раздел практически не ограничен размером, но должен быть поделен на логические

Смещение hex	Размер байт	Содержание
00h	1	ПРИЗНАК ЗАГРУЗКИ 80h – загружаемый раздел 00h – незагружаемый раздел
01h 02h 03h	1 1 1	НАЧАЛО РАЗДЕЛА ДИСКА бит 0 – 7: номер головки (0-255) бит 0 – 5: номер сектора (1-63) бит 6 и 7: старшие биты номера цилиндра бит 0 – 7: младшие биты номера цилиндра (0-1023)
04h	1	ТИП РАЗДЕЛА 00 – раздел не используется 01h – DOS 2.X с 12-битовой FAT 04h – DOS 3.X с 16-битовой FAT 05h – DOS 3.30 Extended-раздел 06h – DOS 4.X с 16-битовой FAT
05h 06h 07h	1 1 1	КОНЕЦ РАЗДЕЛА ДИСКА бит 0 – 7: номер головки (0-255) бит 0 – 5: номер сектора (1-63) бит 6 и 7: старшие биты номера цилиндра бит 0 – 7: младшие биты номера цилиндра (0-1023)
08h	4	ОТНОСИТЕЛЬНЫЙ СЕКТОР Количество секторов перед началом раздела
0Ch	4	РАЗДЕЛ Количество секторов в разделе

Рис.3. Формат полей описания раздела диска.

диски (D,E,F и т.д.) с объемом каждого не более 32 Мбайтов.

"Во всех частях земного шара имеются свои, даже иногда очень любопытные, другие части".

Если поле "Тип раздела" содержит код 05h (Extended-раздел), то физический сектор, определяемый в поле "Начало раздела диска", является вовсе не Boot-сектором Extended-раздела, а вторичным Master Boot-сектором винчестера (Secondary Boot Sector). Этот сектор содержит собственную таблицу разделов, называемую Таблицей Логического Диска (Logical Drive Table), и непременную сигнатуру 0AA55h. Эта таблица и определяет местоположение и размер раздела, с которым DOS, вообще говоря, обращается, как с отдельным физическим диском. Вторичный Master Boot-сектор отличается от Master Boot-сектора, во-первых, тем, что он не содержит программы IPL1 и, соответственно, никогда не определяет загрузочный диск. Во-вторых, Таблица Логического Диска содержит максимум два 16-байтных элемента, а не четыре, как у Partition Table. Причем, если поле "Тип раздела" первого элемента таблицы определяет версию DOS, то такое же поле второго элемента таблицы (если он существует) содержит код Extended-раздела — 05h. Таким образом, второй элемент Таблицы Логического Диска определяет следующий вторичный Master Boot-сектор и т.д. (рис. 4). Каждый диск, определяемый Таблицей Логического Диска, честно содержит Boot-сектор, две копии FAT, корневую директорию, безусловно, область данных и форматируется посредством DOS. Понятно, что расположение Boot-сектора логического диска определяется первым 16-байтным элементом Таблицы.

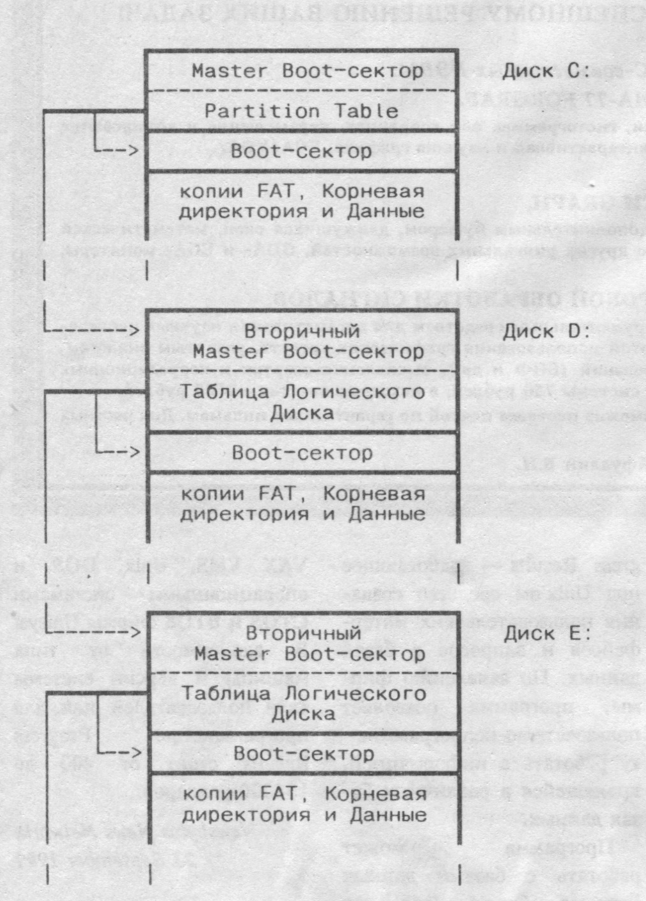


Рис.4. Формирование логических дисков винчестера.

"Где начало того конца, которым оканчивается начало?"

Число секторов до начала раздела хранится в 4-байтном поле "Относительный сектор". Это число определяется путем последовательного подсчета секторов, начиная с сектора 1, головки 0, цилиндра 0 физического диска, и увеличения номера сектора на дорожке, затем номера головки, затем цилиндра. Число секторов в разделе хранится в 4-байтном поле "Размер". Как и для предыдущего поля, первое слово содержит младшую часть числа, второе — старшую.

"Что имеем не храним; потерявши — плачем".

Нет нужды говорить о важности информации, хранящейся в Master Boot- и Boot-секторах винчестера. Сохранение копий этих секторов в виде файлов на дискете — не пустая предосторожность. Если верить Д.Н.Лозинскому, "наша страна выходит на первое место в мире по производству вирусов" (хоть в чем-то впереди!). Уже сейчас известны отечественные "продукты", портящие содержание BPB или Partition Table. Воспользовавшись утилитой NU (Norton Utilities) или специальной программой типа SAV_BOOT (CGT Associates), вы избавите себя от возможных неприятных хлопот.

"И при железных дорогах лучше сохранять двуколку".

Не огорчайтесь, если под рукой нет ничего подходящего. Для получения копий секторов понадобится только утилита DOS DEBUG. Создайте в текстовом редакторе два маленьких файла — M_BOOT.DBG:

```
a 100
mov    bx,200
mov    ax,201
mov    cx,1
mov    dx,80
int    13
int    3
```

```
« здесь пустая строка »
g
rcx
200
n a:\master.sec
w cs:200
q
```

и BOOT.DBG:

```
l cs:100 2 0 1
rcx
200
n a:\boot.sec
w
q
```

соответственно. После выполнения команд

```
DEBUG < M_BOOT.DBG
DEBUG < BOOT.DBG
```

содержание Master Boot- и Boot-секторов будет сохранено на дискете в файлах MASTER.SEC и BOOT.SEC. Использование int 3 позволяет проверить, насколько удачно завершилось чтение сектора (отсутствие признака переноса и нулевое значение регистра AH).

"Плюнь тому в глаза, кто скажет, что можно объять необъятное!"

В последнее время для конфигурации винчестера вместо утилиты FDISK часто используют специальные программы — "диск-менеджеры". Среди них широко распространены такие программы, как Disk Manager, Advanced Disk Manager, Speed Stor. Они имеют собственные таблицы разделов, свои драйверы дисковых устройств, причем необходимая служебная информация может храниться во втором физическом секторе (сектор 2, головка 0, цилиндр 0). Применение "диск-менеджеров" связано с предоставлением ими таких дополнительных возможностей, как защита логического диска от записи, организация парольной защиты и, конечно, создание логических дисков размером более 32 Мбайтов при работе с DOS 3.30.

Вместе с *Козьмой Прутковым*
составил *А. Борзенко*

Borland начинает поставку утилиты Screenery для Microsoft Windows 3.0. В нее входят 40 типов фоновой картинки для Windows, шесть алгоритмов красивой утилиты гашения экрана и иконки для 50 продуктов. Все это работает с VGA адаптером 640x480. Цена — 35 долларов.

"Даже самые серьезные пользователи могут получать удовольствие от Screenery. Этой программой мы еще раз подтверждаем наше желание не только делать компьютеры более производительными, но и более интересными", — сказал вице-президент Borland Стефан Кан.

Newsbytes News Network,
23 September 1991

Фирма Progress Software выпустила программу Progress Results — работающее под Unix´ом средство создания пользовательских интерфейсов и запросов в базах данных. По заявлению фирмы, программа позволяет пользователю-непрограммисту работать с информацией, хранящейся в различных базах данных.

Программа может работать с базами данных Progress, Oracle, Rdb под VAX VMS, Unix, DOS, и операционными системами CTOS и BTOS фирмы Unisys. В зависимости от типа машины и версии системы (для пользователей или для программистов) Progress Results стоит от 400 до 134000 долларов.

Newsbytes News Network,
23 September 1991

НОВОСТИ

Раздел новостей в этом номере составлен в основном из информации, опубликованной новым электронным бюллетенем новостей The Teleputing Hotline. Подписка на его русскую версию открылась в сети электронной почты RELCOM.

The Teleputing Hotline выходит дважды в неделю (каждый выпуск — около 10 килобайт), редактируется в Атланте (США) и Лондоне, и стоит для абонентов Релкома всего 85 рублей в месяц в отличие от американской версии, продающейся за 50 долл. Русская версия Hotline выходит в свет через пять часов после появления его за океаном.

Как говорят знающие люди, это одно из немногих изданий, в которых можно оперативно узнать, что делают в нашей стране иностранные компьютерные и коммуникационные фирмы. Электронные издания, подобные Hotline, весьма популярны на Западе, но в СССР практически неизвестны.

А еще одним необычным моментом является то, что в течение одного месяца читать бюллетень можно совершенно бесплатно...

Если вы захотите подписаться на The Teleputing Hotline — узлы Релкома есть уже в 80 городах страны — от Калининграда до Владивостока. О них вам вежливо расскажут по телефону в Москве (095)231-21-29. Сетевой адрес редактора русской версии Hotline: kirill@newsbytes.msk.su

ТАЙВАНЬСКАЯ ФИРМА ACER выпустила в продажу в США и Гонконге новый карманный компьютер. К нему можно купить диски большей емкости и модули расширения оперативной памяти. Начальная цена 17000 гонконгских долларов (2000 долларов США). В Соединенных Штатах эти машины будут продаваться через оптовых поставщиков типа Circuit City.

AQUILINE выпустила компьютер-записную книжку весом 7,9 фунта (3,6 кг) с микропроцессором Intel 80386/33 МГц. Стандартный микропроцессор для машин такого класса 80386-SX, работает на частоте до 20 МГц.

BONDWELL представила серию переносных laptop-компьютеров с датчиком перемещений и системой предотвращения неавторизованного использования жесткого диска.

WYSE выпустила компьютер-записную книжку весом 4,5 фунта (2 кг) на базе микропроцессора Intel 80386-SX/20 МГц с белоснежным экраном. Дополнительные модули предусматривают такие возможности, как факс-модем и второй последовательный порт.

Технология едет в “Россию со товарищи“

Пробужденные переворотом в СССР телефонные компании со всего мира стекаются в Союз. Из “списка запрещенных компьютеров“ западных экспортеров исчезла четверть позиций. В настоящее время экспортировать PC в Москву стало легче, чем в Иран, Ирак, Ливию, Сирию или Северную Корею, которые США обвиняют в поддержке терроризма. Оптический кабель, который обеспечил бы защищенный от подслушивания трафик, все еще в списке. US Sprint форсировала планы по установке узлов Sprintnet в 12 российских городах. AT&T имеет 91 спутниковый телефонный канал в СССР, включая 24 через Интерспутник. Ей хотелось бы получить еще 42, вдобавок к 67, через западную спутниковую сеть Intelsat. MCI и US Sprint тоже могут попросить ломтик от этого пирога.

US West, которая была в России раньше всех, заявила что она может предложить сотовую телефонную сеть для Ленинграда через несколько недель. Эта система работает на частоте 450МГц, а не на стандартной на Западе частоте 900МГц, так как высокие частоты заняты военными. Местные жители будут платить в рублях, а иностранцы — в твердой валюте. Компания запустит свою систему в Москве во втором квартале 1992 года, а еще через 18 месяцев установит свои коммутаторы в 3 русских городах для того, чтобы упростить международные звонки.

Американские фирмы делают свой ход в войне “записных книжек“

Американские компании выпустили очередной залп компьютеров класса записных книжек (notebook). Планка, стоящая перед соревнующимися — 3,9 кг (7 фунтов) веса, 80 Мбайтные жесткие диски, дисплеи с изображением “черное-на-белом“, батареи с большим сроком работы, 4 Мбайта оперативной памяти — и все это дешевле, чем за 5000 долларов.

ФИРМА Grid добавила к своей записной книжке 1750 с процессором 80386SX операционную систему MS-DOS 5.0 и 1 мегабайт памяти, расширяемый до пяти. Фирма отмечает, что DOS 5 облегчает использование этой памяти. Розничные цены на изделие объявлены не были — признак того, что они быстро упадут. Dell предлагает System 320N+, “записную книжку“ с процессором 80386SX, оперативной памятью до 8 Мбайтов, жесткими дисками емкостью до 80 Мбайтов и никель-гидридовыми батареями, работающими от одной зарядки 4-5 часов. Цены на более старую серию 320N были снижены и теперь не превышают 2500 долларов. Компания AT&T также добавила жесткий диск на 80 Мбайтов к своей Safari 80386SX и снизила на 12% цены на более старую модель с 40 Мбайтами. В новой модели Safari также имеется встроенный факс-модем на 2400 бод (4 страницы текста в

минуту), работающий с сотовыми телефонами. Цена этого компьютера, однако, 5399 долларов, и весит он 4,1 кг. Фирма AST подняла скорость компьютеров этого класса до 25 МГц на 80386, используя микросхему Advanced Micro Devices. Новое изделие выпускается вместе с DOS 5, экраном "черное-на-белом", 4 Мбайтами памяти и жестким диском на 80 Мбайтов по цене 4795 долларов. Батареи работают 3 часа.

Фирма Librex — подразделение корпорации Nippon Steel — ответила еще одним снижением цен на целых 43% на модели, использующие процессор 80286. Цена, установленная этой фирмой, на модель с процессором 80386SX, 4 Мбайтами памяти и жестким диском на 20 Мбайтов равна сейчас 2999 долларов.

КОМПАНИЯ SHARP заявила, что ею разработан цветной дисплей на жидких кристаллах толщиной 12 мм, потребляющий в 3 раза меньше электроэнергии по сравнению с текущими моделями.

ФИРМА KDD подписала соглашение с узбекским Министерством связи об установлении 30 международных спутниковых линий между Ташкентом и Токио следующим летом. KDD также уже заключила сделку по предоставлению услуг во Владивостоке и на острове Сахалин.

US WEST открыла свою сотовую систему, работающую на частоте 450 МГц, под названием Delta Telecom в Санкт-Петербурге, Россия. Эта фирма также открыла системы с частотой 950 МГц в Праге через государственное предприятие Eurotel. Компания также примет участие в испытаниях цифровой сотовой технологии CDMA фирмы Qualcomm. Американской сотовой индустрией была принята технология TDMA, но использование CDMA весьма вероятно в микроволновых службах PCN.

SILVERWARE анонсировала SilverFox SPCS и SilverClip SPCS — инструментальный пакет программ, с помощью которых можно перевести базы данных Clipper 5.01 и FoxPro 2.0 в режим on-line.

ЯПОНИЯ начинает новый проект стоимостью 88,9 миллионов долларов для развития коммерческой спутниковой технологии. Проект на одну треть финансируется правительством.

Модемы v.32/9600 бод становятся настоящими предметами массового потребления

С наступлением осени стали падать не только листья, но и цены на модемы. Продажные цены на модемы со скоростью передачи 9600 бод, отвечающие стандарту V.32, к концу года должны упасть ниже 300 долларов. Со сжатием данных такие модемы могут передавать информацию со скоростью 37600 бит в секунду; это означает, что с их помощью можно эффективно пересылать графические и музыкальные файлы. Это многое обещает сообразительным операторам электронных бюллетеней и бросает вызов операторам пакетных сетей типа CompuServe, Sprintnet и Tymnet. Им нужно быстро переходить на работу со скоростью 9600 бод, иначе они проиграют в сравнении с прямым телефонным звонком. Такая ситуация может быстро привести к разгрому фирмы CompuCom, производящей модем на 9600 бод ценой 169 долларов, — "чемпион по скорости" среди дешевых моделей, но не отвечающий стандарту V.32. Покупатели модемов смогут увеличить скорость в 4 раза, а эффективную скорость (с учетом сжатия) в 16 раз за те же деньги, которые они платили несколько лет назад за изделия со скоростью 2400 бод.

КОМПАНИЯ HAYES, лидер на рынке, предлагает "глобальную ценовую стратегию", отменяя ранее предлагавшиеся розничные цены на свою продукцию и гарантируя, что встретит конкурентов во всеоружии. Первый с момента действия новой стратегии модем, представленный в Лондоне, — модем V.32 Optima. Он не связан с существующей серией V.29 V-модемов этой фирмы. Dennis Hayes, по сообщению Steve Gold, назвал розничные цены "искусственно получаемой цифрой, которая часто вызывает недоразумения с ценами на рынке". Реальные цены, по которым происходит продажа, часто ниже на сотни долларов (или фунтов) даже на продукцию Hayes. Ожидается, что реальная цена, по которой в настоящее время продается Hayes Ultra 9600 — наиболее дорогое изделие фирмы — упадет с 1000 фунтов стерлингов до 800 в течение нескольких недель.

По крайней мере одна конкурирующая компания сообщила, что она создаст новую фирму и изделие с новым торговым названием, чтобы ее дешевые модемы V.32 не "вторгались" в уже существующие серии моделей. Компания Phylon, которая поставляет фирме Hayes микросхемы, представила новую версию своего комплекта микросхем на 9600 бод по цене 50 долларов при продаже партиями.

Другой вариант ответа на складывающуюся ситуацию — ускоряться. Новые комплекты микросхем фирмы Rockwell работают в соответствии со стандартом V.32 bis со скоростью 14400 и продаются партиями. Фирма Telebit будет использовать их для производства T3000, — изделия, работающего по стандарту V.32 bis со скоростью 14400 и обладающего интерфейсом, который со сжатием данных позволяет достичь производительности 57600 бит в секунду. Такой модем мог бы заменить арендуемые линии со скоростью передачи 56000 бит в секунду. В стандартную модель, продаваемую по розничной цене 1095 долларов, включены также средства защиты информации.

Советская электронная почта расстроила планы заговорщиков

Кирилл Чащин сообщает из России, что RELCOM — система электронной почты — была важным звеном в цепи обстоятельств, остановивших переворот в августе. Свидетельства очевидцев разлетались по сети прежде, чем ГКЧП мог бы сообщить о своих действиях, и мгновенно читались 7 тысячами пользователей. За считанные часы "Интерфакс" и Российское Информационное Агентство передавали свои сообщения по сети RELCOM, в том числе и за рубеж, а к ним по сети приходила информация из первых рук. На основе этих сообщений работали бесчисленные западные информационные агентства.

Очередной срыв в работе Нью-Йоркской службы AT&T

Сбой в системе энергоснабжения в коммутационном центре в нижнем Манхеттене отключил междугородное телефонное сообщение с Нью-Йорком, причем нельзя было позвонить ни в город, ни из города. Это произошло 17 сентября в 4 часа 50 минут вечера по Восточному времени (часовой пояс атлантического побережья). В полной мере работа была восстановлена только к полуночи.

Эта неприятность задержала авиарейсы во всех трех нью-йоркских аэропортах. Операторы AT&T получили инструкции давать клиентам междугородные коды конкурирующих компаний. Это третья крупная авария в AT&T менее, чем за 2 года. В январе 1990 года сбой в работе компьютерной программы вызвал цепную реакцию, которая отключила почти полностью всю телефонную сеть, а в январе 1991 года разрыв телефонного кабеля между Нью-Йорком и Ньюмарком, штат Нью-Джерси, также привел к прекращению работы.

В персональном компьютере palmtop ("с ладонь") фирмы SONY обнаружены дефекты

Sony прекратила поставку своих компьютеров размером с ладонь (palmtop) PTC300 из-за дефекта во встроенном программном обеспечении. Когда машине предлагается выполнить функции, не описанные в руководстве, она зависает или перестает работать клавиатура. Если это произошло, пользователю ничего не остается, кроме как нажать кнопку "сброс" и расстаться с результатами текущей работы. Одной из главных привлекательных черт устройства является возможность ввода рукописной информации. PTC300 стоит 65000 йен (480 долларов); это примерно одна треть от стоимости его предшественника PTC500, выпущенного той же фирмой.

На этой странице помещен бланк заказа на сборник «КомпьютерПресс»
Вы можете его вырезать и, заполнив, отправить в конверте по адресу:
113093, Москва, а/я 37.

Подписка на 1992 г. принимается до 31 января 1992 г. Число экземпляров — без ограничений.

Вы можете выписать журнал на полгода или на год. Стоимость годовой подписки на «КомпьютерПресс» — 57 рублей 60 копеек.

Деньги следует перечислить на расчетный счет агентства "КомпьютерПресс".

Банковские реквизиты:
получатель: Автобанк (для зачисления на счет №345708)
расчетный счет получателя: №161202
банк получателя: ЦОУ при Госбанке СССР. МФО №299112.
Копию платежного документа необходимо приложить к бланку заказа.

Без одновременной оплаты подписной стоимости заказ не принимается. Издания агентства "КомпьютерПресс" наложенным платежом не высылаются.

ЗАКАЗ

От кого ______________________________

Адрес ______________________________
(ПОЧТОВЫЙ ИНДЕКС УКАЗЫВАТЬ ОБЯЗАТЕЛЬНО)

Прошу оформить подписку на 1992 год

Подписная плата в сумме ______________ перечислена

платежным поручением (почтовым переводом) № _______ от _______________199_ г.

(Копия платежного документа прилагается)

Заказ

Советско-американское предприятие "Соваминко"

Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 191186, Ленинград, Невский проспект, 28, Магазин № 1 "Дом книги"

От кого . . . .

Адрес . . . .
(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

Заказ

Советско-американское предприятие "Соваминко"

Рекламно-издательское агентство "КомпьютерПресс"

Принимает заказы на журнал "КомпьютерПресс" и производит отправку наложенным платежом.

Заказ высылается по адресу: 630076, Новосибирск, Красный проспект, 60 Магазин № 7 "Техническая книга"

Телефон для справок: 20-05-09

От кого . . . .

Адрес . . . .
(почтовый индекс указывать обязательно)

. . . .

Номера выпусков . . . . **Количество экземпляров** . . . .

Цена 3.15